CLEV-1

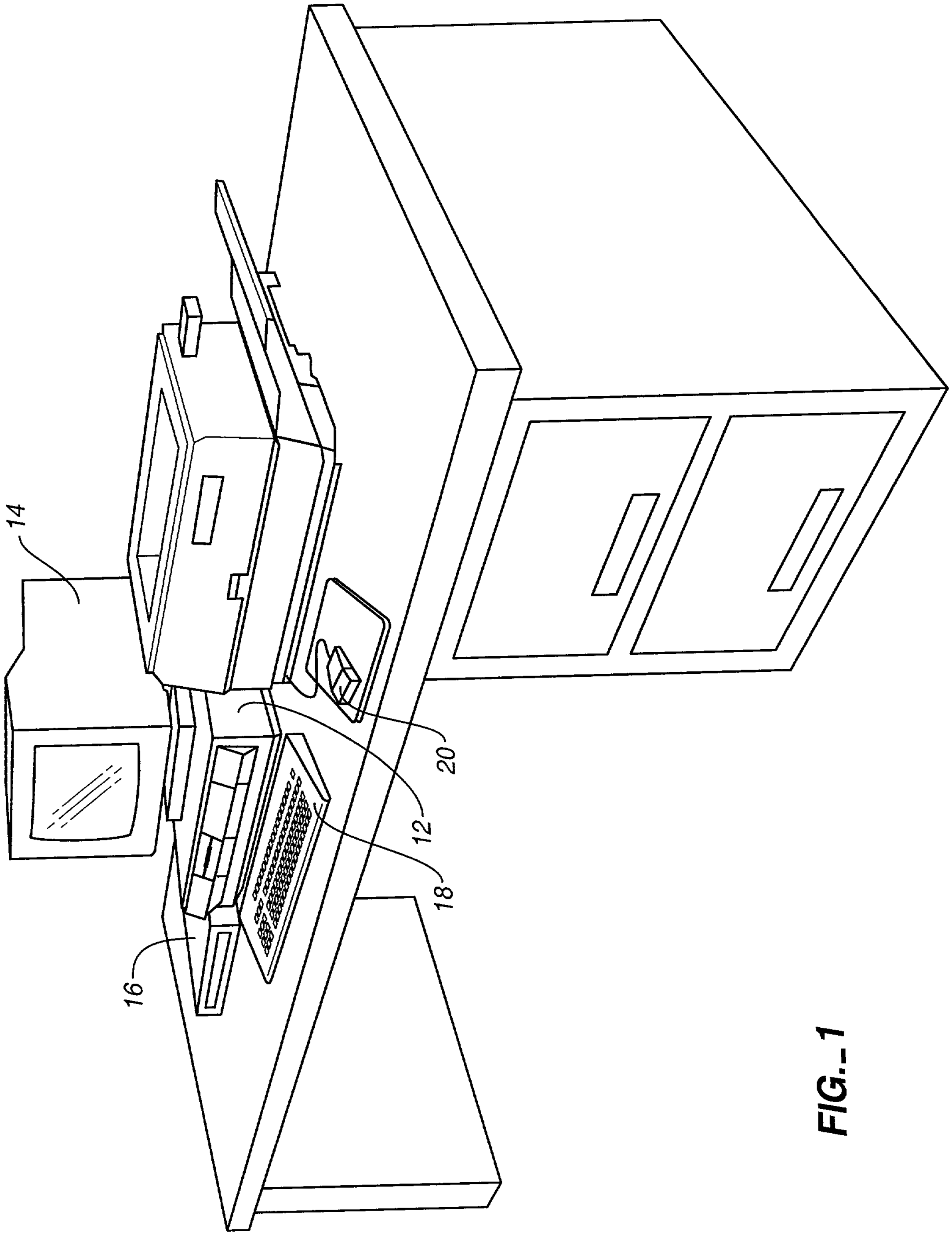

FIG._1

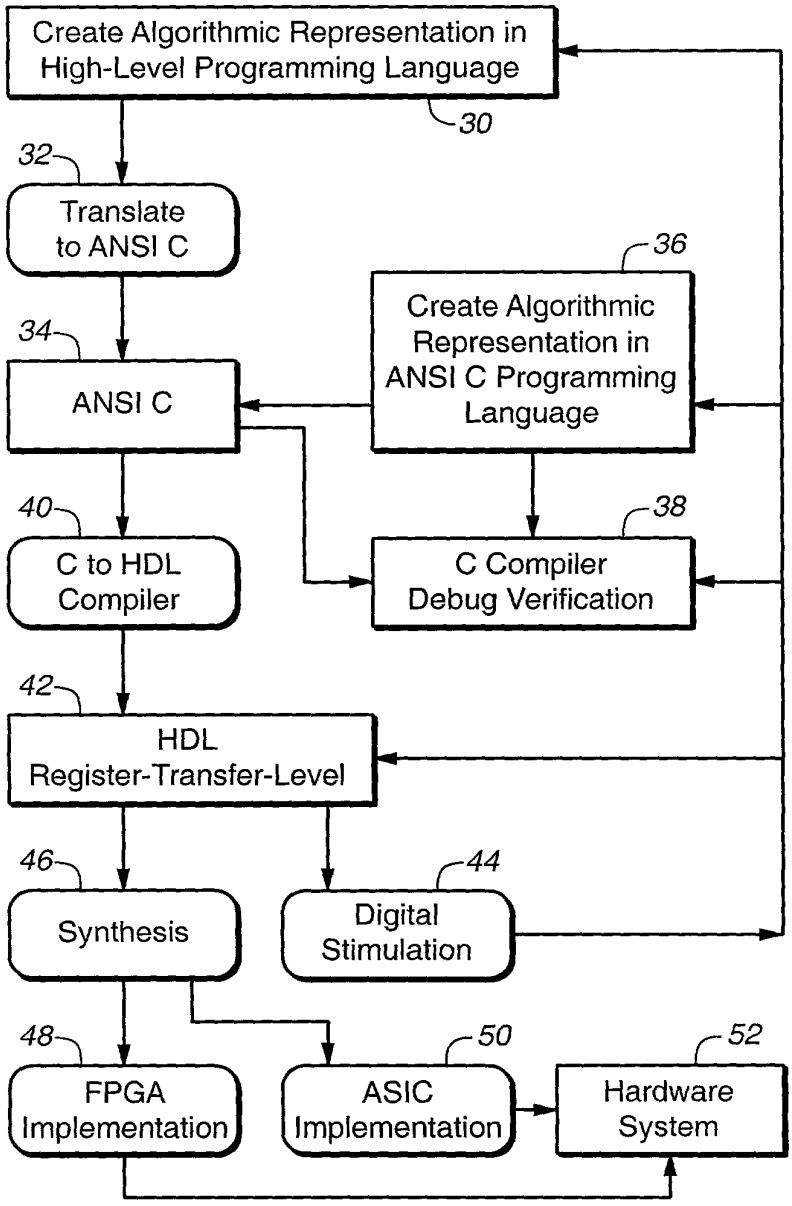

FIG._2

```
int func1 (unsigned a, unsigned b, unsigned c)
{
   return a + b ^ c << 3;
}

int func2 (int a, int b, int c)
{
   return (a ^ b) & c;
}
```

FIG._3A

```
module func1(result_func1, result_func2, param_func1_a, param_func1_b,
   param_func1_c, param_func2_a, param_func2_b, param_func2_c);

  output [15:0]    result_func1, result_func2;
  input  [15:0]    param_func1_a, param_func1_b, param_func1_c;
  input  [15:0]    param_func2_a, param_func2_b, param_func2_c;

  wire   [15:0]    result_func1 =
                   ((param_func1_a + param_func1_b) ^ (param_func1_c << 16'd3));
  wire   [15:0]    result_func2 =
                   ((param_func2_a ^ param_func2_b) & param_func2_c);

endmodule
```

FIG._3B

```
int func1 (unsigned a, unsigned b, unsigned c)
{
   return a + (b == 1) ^ c << 3;
}

int func2 (int a, int b, int c)
{
   if (a < b)
      c -= b;

   return (a ^ b) & c;
}

int func3 (unsigned a, unsigned b, int c)
{
   unsigned d = 0;
   unsigned e;

   b++;

   if (a < b)
   {
      int c = e = a - b;

      if (a != 0)
      {
         c -= a & b;
         d = c++ ^ b;
      }
      else
      {
         d++;
         e = c + d ? c : 1;
      }
   }
   d = 1;
   return a < b && (a + b ^ c != 3 || a - b == 0);
}
```

FIG._4A

```
module funcx (clock,result_func1,result_func2,result_func3,param_func1_a,
        param_func1_b,param_func1_c,param_func2_a,param_func2_b,
        param_func2_c,param_func3_a,param_func3_b,param_func3_c);
   input          clock;
   input  [15:0]  param_func1_a, param_func1_b, param_func1_c, param_func2_a, param_func2_b,
                  param_func2_c, param_func3_a, param_func3_b, param_func3_c;
   output [15:0]  result_func1, result_func2, result_func3;
   reg            _5_1;
   reg    [15:0]  result_func1, result_func2, result_func3,
                  _func2_c, _func3_b, _6_func3_d, _6_func3_e,
                  _7_func3_c, _1_16, _4_16, _6_16, _7_16;
   reg    [16:0]  _2_17, _3_17;
always @(posedge clock)
   begin
     _1_16 = (param_func1_b == 16'd1);
     result_func1 = ((param_func1_a + _1_16) ^ (param_func1_c << 16'd3));
     _func2_c = param_func2_c;
     _2_17 = { ~ param_func2_a [15], param_func2_a };
     _3_17 = { ~ param_func2_b [15], param_func2_b };
     if ((_2_17 < _3_17))
        _func2_c = (_func2_c - param_func2_b);
     result_func2 = ((param_func2_a ^ param_func2_b) & _func2_c);
     _func3_b = param_func3_b;
     _6_func3_d = 16'd0;
     _func3_b = (_func3_b + 16'd1);
     if ((param_func3_a < _func3_b))
      begin
       _6_func3_e = (param_func3_a - _func3_b);
       _7_func3_c = _6_func3_e;
       if ((param_func3_a != 16'd0))
        begin
         _7_func3_c = (_7_func3_c - (param_func3_a & _func3_b));
         _4_16 = _7_func3_c;
         _7_func3_c = (_4_16 + 16'd1);
         _6_func3_d = (_4_16 ^ _func3_b);
        end
        else
        begin
         _6_func3_d = (_6_func3_d + 16'd1);
         _6_func3_e = ((_7_func3_c + _6_func3_d) ? _7_func3_c : 16'd1);
        end
      end
     _6_func3_d = 16'd1;
     _6_16 = (param_func3_c != 16'd3);
     _5_1  = (param_func3_a < _func3_b) && (((param_func3_a + _func3_b) ^ _6_16)
                || ((param_func3_a - _func3_b) == 16'd0));
      _7_16 = _5_1;
     result_func3 = _7_16;
   end
endmodule
```

FIG._4B

```
int sum1 (int n)
{
   int i, sum = 0;
   for (i = 0; i < n; i ++)
      sum += i;
   return sum;
}

int sum2 (int array [], int size)
{
   int i, sum = 0;
   for (i = 0; i < size; i ++)
      sum += array [i];
   return sum;
}

int main ()
{
   int i;
   int array [10];
   int size = sizeof (array) / sizeof (*array);
   for (i = 0; i < size; i++)
      array [i] = i * 2;
   return sum1 (size) + sum2 (array, size);
}
```

FIG._5A

```
`define _5_main_array 4'h1

module RAM32X1 (O, D, WE, A4, A3, A2, A1, A0);
 /* synthesis black_box xc_alias="RAM" */
   input          D, WE, A4, A3, A2, A1, A0;
   output         O;
   reg [31:0]     d;
   wire [4:0]     A = { A4, A3, A2, A1, A0 };
   assign         O = d [A];
   always @(A or D or WE)
      if (WE)
         d [A] = D;
endmodule

module RAM32x1 (a, d, o, we);
   input  [4:0]   a;
   input          d, we;
   output         o;
   RAM32X1 RAM32X1 (o, d, we, a [4], a [3], a [2], a [1], a [0]);
endmodule

module RAM32x16 (a, d, o, we);
   input          we;
   input  [ 4:0]  a;
   input  [15:0]  d;
   output [15:0]  o;
   RAM32x1 U0  (a, d [ 0], o [ 0], we);
   RAM32x1 U1  (a, d [ 1], o [ 1], we);
   RAM32x1 U2  (a, d [ 2], o [ 2], we);
   RAM32x1 U3  (a, d [ 3], o [ 3], we);
   RAM32x1 U4  (a, d [ 4], o [ 4], we);
   RAM32x1 U5  (a, d [ 5], o [ 5], we);
   RAM32x1 U6  (a, d [ 6], o [ 6], we);
   RAM32x1 U7  (a, d [ 7], o [ 7], we);
   RAM32x1 U8  (a, d [ 8], o [ 8], we);
   RAM32x1 U9  (a, d [ 9], o [ 9], we);
   RAM32x1 U10 (a, d [10], o [10], we);
   RAM32x1 U11 (a, d [11], o [11], we);
   RAM32x1 U12 (a, d [12], o [12], we);
   RAM32x1 U13 (a, d [13], o [13], we);
   RAM32x1 U14 (a, d [14], o [14], we);
   RAM32x1 U15 (a, d [15], o [15], we);
endmodule

module RAM11x16 (a, d, o, we);
   input          we;
   input  [ 3:0]  a;
   input  [15:0]  d;
   output [15:0]  o;
   RAM32x16 U ({ 1'b0, a }, d, o, we);
endmodule
```

FIG._5B1

```
module for1 (clock, reset, result_sum1, result_sum2, result_main,
        param_sum1_n, param_sum2_array, param_sum2_size, run_sum1,
        ready_sum1, run_sum2, ready_sum2, run_main, ready_main);

   input            clock, reset, run_sum1, run_sum2, run_main;
   input   [15:0]   param_sum1_n, param_sum2_array, param_sum2_size;
   output           ready_sum1, ready_sum2, ready_main;
   output  [15:0]   result_sum1, result_sum2, result_main;
   reg     [15:0]   result_sum1, result_sum2, result_main;
   reg              ready_sum1, ready_sum2, ready_main, we;
   wire    [15:0]   o, d;
   reg     [ 3:0]   state, a, return_state_sum1, return_state_sum2;
   reg     [15:0]   _sum1_n, _2_sum1_i, _2_sum1_sum, _sum2_array, _sum2_size;
   reg     [15:0]   _4_sum2_i, _4_sum2_sum, _5_main_i, _5_main_size, _5_16, _8_16;
   reg     [16:0]   _1_17, _2_17, _3_17, _4_17, _6_17, _7_17;

   RAM11x16 m (a, d, o, we);
   parameter _sum1 = 1, _sum2 = 5;

   always @(posedge clock)
   begin
     if (reset)
     begin
         we = 0;
         state = 0;
     end
     else
     begin
       case (state)
     0: begin
         if (run_sum1)
         begin
           ready_sum1 = 0;
           return_state_sum1 = 0;
           _sum1_n = param_sum1_n;
           state = _sum1;
         end
         else if (run_sum2)
         begin
           ready_sum2 = 0;
           return_state_sum2 = 0;
           _sum2_array = param_sum2_array;
           _sum2_size = param_sum2_size;
           state = _sum2;
         end
         else if (run_main)
         begin
           ready_main = 0;
           _5_main_size = 16'd10;
           _5_main_i = 16'd0;
           state = 9;
         end
       end
```

FIG._5B2

```
_sum1: begin
    _2_sum1_sum = 16'd0;
    _2_sum1_i = 16'd0;
    state = 2;
  end
2: begin
    _1_17 = { ~ _2_sum1_i [15], _2_sum1_i };
    _2_17 = { ~ _sum1_n [15], _sum1_n };
    state = (_1_17 < _2_17) ? 3 : 4;
  end
3: begin
    _2_sum1_sum = (_2_sum1_sum + _2_sum1_i);
    _2_sum1_i = (_2_sum1_i + 16'd1);
    state = 2;
  end
4: begin
    result_sum1 = _2_sum1_sum;
    ready_sum1 = 1;
    state = return_state_sum1;
  end
sum2: begin
    _4_sum2_sum = 16'd0;
    _4_sum2_i = 16'd0;
    state = 6;
  end
6: begin
    _3_17 = { ~ _4_sum2_i [15], _4_sum2_i };
    _4_17 = { ~ _sum2_size [15], _sum2_size };
    state = (_3_17 < _4_17) ? 7 : 8;
  end
7: begin
    a = (_sum2_array + _4_sum2_i);
    state = 12;
  end
12: begin
    _5_16 = o;
    _4_sum2_sum = (_4_sum2_sum + _5_16);
    _4_sum2_i = (_4_sum2_i + 16'd1);
    state = 6;
  end
8: begin
    result_sum2 = _4_sum2_sum;
    ready_sum2 = 1;
    state = return_state_sum2;
  end
9: begin
    _6_17 = { ~ _5_main_i [15], _5_main_i };
    _7_17 = { ~ _5_main_size [15], _5_main_size };
    state = (_6_17 < _7_17) ? 10 : 11;
  end
```

FIG._5B3

```
10: begin
     _8_16 = (_5_main_i << 1'd1);
     a = (`_5_main_array + _5_main_i);
     d = _8_16;
     we = 1;
     state = 13;
   end
13: begin
     we = 0;
     _5_main_i = (_5_main_i + 16'd1);
     state = 9;
   end
11: begin
     _sum1_n = _5_main_size;
     return_state_sum1 = 14;
     state = _sum1;
   end
14: begin
     _sum2_array = `_5_main_array;
     _sum2_size = _5_main_size;
     return_state_sum2 = 15;
     state = _sum2;
   end
15: begin
     result_main = (result_sum1 + result_sum2);
     ready_main = 1;
     state = 0;
   end
default: ;
   endcase
  end
 end
endmodule
```

FIG._5B4

```
#include <stdio.h>

void main ()
{
   int a, b;
   scanf ("%d %d", & a, & b);
   printf ("a + b = %d\n", a + b);
}
```

FIG._6A

```
module Printf (clock, scanf_0_1_line_19_a, scanf_0_2_line_19_b, printf_1_1_line_20, run_main);
   input          clock, run_main;
   input  [15:0]  scanf_0_1_line_19_a, scanf_0_2_line_19_b;
   output [15:0]  printf_1_1_line_20;
   reg    [15:0]  printf_1_1_line_20, _3_main_a, _3_main_b;

   always @(posedge clock)
     begin
      if (run_main)
      begin
         _3_main_a = scanf_0_1_line_19_a;
         _3_main_b = scanf_0_2_line_19_b;
         printf_1_1_line_20 = (_3_main_a + _3_main_b);

         // User Verilog code
         $write ("a + b = %d\n", printf_1_1_line_20);
         // End of user Verilog code
      end
   end
endmodule
```

FIG._6B

```
int func (a, b, c, d)
{
   return (a / b + c / d);
}
```

FIG._7A

```
module udivmod16 (a, b, div, mod);
   input  [15:0]  a, b;
   output [15:0]  div, mod;
   reg    [15:0]  div, mod, pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7, pa8, pa9, pa10, pa11, pa12,
                  pa13, pa14, pa15, pa16, shb0;
   reg    [16:0]  shb1;
   reg    [17:0]  shb2;
   reg    [18:0]  shb3;
   reg    [19:0]  shb4;
   reg    [20:0]  shb5;
   reg    [21:0]  shb6;
   reg    [22:0]  shb7;
   reg    [23:0]  shb8;
   reg    [24:0]  shb9;
   reg    [25:0]  shb10;
   reg    [26:0]  shb11;
   reg    [27:0]  shb12;
   reg    [28:0]  shb13;
   reg    [29:0]  shb14;
   reg    [30:0]  shb15;

   always @(a or b)
   begin
      pa0 = a;

      shb15   = b << 15;
      div [15] = pa0 >= shb15;
      pa1     = div [15] ? pa0 - shb15 : pa0;

      shb14   = b << 14;
      div [14] = pa1 >= shb14;
      pa2     = div [14] ? pa1 - shb14 : pa1;

      shb13   = b << 13;
      div [13] = pa2 >= shb13;
      pa3     = div [13] ? pa2 - shb13 : pa2;
```

FIG._7B1

```
    shb12    = b << 12;
    div [12] = pa3 >= shb12;
    pa4      = div [12] ? pa3 - shb12 : pa3;

    shb11    = b << 11;
    div [11] = pa4 >= shb11;
    pa5      = div [11] ? pa4 - shb11 : pa4;

    shb10    = b << 10;
    div [10] = pa5 >= shb10;
    pa6      = div [10] ? pa5 - shb10 : pa5;

    shb9     = b << 9;
    div [ 9] = pa6 >= shb9;
    pa7      = div [9] ? pa6 - shb9 : pa6;

    shb8     = b << 8;
    div [ 8] = pa7 >= shb8;
    pa8      = div [8] ? pa7 - shb8 : pa7;

    shb7     = b << 7;
    div [ 7] = pa8 >= shb7;
    pa9      = div [7] ? pa8 - shb7 : pa8;

    shb6     = b << 6;
    div [ 6] = pa9 >= shb6;
    pa10     = div [6] ? pa9 - shb6 : pa9;

    shb5     = b << 5;
    div [ 5] = pa10 >= shb5;
    pa11     = div [5] ? pa10 - shb5 : pa10;

    shb4     = b << 4;
    div [ 4] = pa11 >= shb4;
    pa12     = div [4] ? pa11 - shb4 : pa11;

    shb3     = b << 3;
    div [ 3] = pa12 >= shb3;
    pa13     = div [3] ? pa12 - shb3 : pa12;

    shb2     = b << 2;
    div [ 2] = pa13 >= shb2;
    pa14     = div [2] ? pa13 - shb2 : pa13;

    shb1     = b << 1;
    div [ 1] = pa14 >= shb1;
    pa15     = div [1] ? pa14 - shb1 : pa14;

    shb0     = b << 0;
    div [ 0] = pa15 >= shb0;
    pa16     = div [0] ? pa15 - shb0 : pa15;

    mod = pa16;
  end

endmodule
```

FIG._7B2

```
module sdivmod16 (a, b, div, mod);

   input  [15:0]  a, b;
   output [15:0]  div, mod;

   wire   [15:0]  ua  =  a [15] ? - a : a;
   wire   [15:0]  ub  =  b [15] ? - b : b;
   wire   [15:0]  udiv, umod;

   udivmod16 udivmod (ua, ub, udiv, umod);

   wire   [15:0]  div = (a [15] !=  b [15]) ? - udiv : udiv;
   wire   [15:0]  mod =  a [15]  ? - umod : umod;

endmodule



module DivEx (clock,reset,result_func,param_func_a,param_func_b,
                  param_func_c,param_func_d,run_func,ready_func);

   input          clock, reset, run_func;
   input  [15:0]  param_func_a, param_func_b, param_func_c, param_func_d;
   output         ready_func;
   output [15:0]  result_func;

   reg    [15:0]  result_func, divmod_a, divmod_b, _1_16, _2_16;
   reg            ready_func;
   reg    [ 1:0]  state;
   wire   [15:0]  div_result;

   sdivmod16 sdivmod (divmod_a, divmod_b, div_result, );

   always @(posedge clock)
   begin
      if (reset)
         state = 0;
      else
      begin
         case (state)

         0: begin
            if (run_func)
            begin
               ready_func = 0;
               divmod_a = param_func_a;
               divmod_b = param_func_b;
                state = 1;
            end
         end
```

FIG._7B3

```
            1: begin
               _1_16 = div_result;
              divmod_a = param_func_c;
              divmod_b = param_func_d;
               state = 2;
            end

            2: begin
               _2_16 = div_result;
               result_func = (_1_16 + _2_16);
              ready_func = 1;
               state = 0;
            end

            default:  ;

           endcase
        end
    end

endmodule
```

FIG._7B4

```
int data_in;
int out1;
int out2;
int data_out;

void pipeline_stage_1 ()
{
   out1 = data_in + 1;
}

void pipeline_stage_2 ()
{
   out2 = out1 + out1;
}

void pipeline_stage_3 ()
{
   data_out = out2 ^ 1234;
}

#ifndef __SYNETRY__
#include <stdio.h>

void main ()
{
   for (;;)
   {
      scanf ("%d", & data_in);

      pipeline_stage_1 ();
      pipeline_stage_2 ();
      pipeline_stage_3 ();

      printf ("%d", data_out);
   }
}
#endif
```

FIG._8A

```
module Pipeline (clock, _data_in, _data_out, run_pipeline_stage_1,run_pipeline_stage_2,
run_pipeline_stage_3);
   input          clock, run_pipeline_stage_1, run_pipeline_stage_2, run_pipeline_stage_3;
   input  [15:0]  _data_in;
   output [15:0]  _data_out;
   reg    [15:0]  _data_out, _out1, _out2;

   always @(posedge clock)
   begin
     if (run_pipeline_stage_1)
       _out1 = (_data_in + 16'd1);
   end

   always @(posedge clock)
   begin
     if (run_pipeline_stage_2)
       _out2 = (_out1 + _out1);
   end

   always @(posedge clock)
   begin
     if (run_pipeline_stage_3)
       _data_out = (_out2 ^ 16'd1234);
   end
endmodule
```

FIG._8B

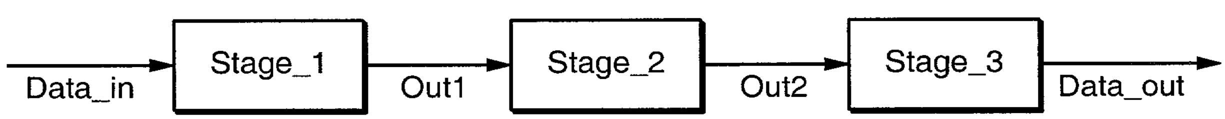

FIG._8C

```
int func (float a, double b, long double c, int i)
{
   return - a + b * c - 16.5 / c;
}
```

FIG._9A

```
module Float (clock,reset,zero_substractor_float_operand,
      zero_substractor_float_run,zero_substractor_float_ready,zero_substractor_float_result,
      convertor_from_float_to_long_double_operand,convertor_from_float_to_long_double_run,
      convertor_from_float_to_long_double_ready,convertor_from_float_to_long_double_result,
      convertor_from_double_to_long_double_operand,convertor_from_double_to_long_double_run,
      convertor_from_double_to_long_double_ready,convertor_from_double_to_long_double_result,
      multiplier_long_double_a,multiplier_long_double_b,multiplier_long_double_run,
      multiplier_long_double_ready,multiplier_long_double_result,adder_long_double_a,
      adder_long_double_b,adder_long_double_run, adder_long_double_ready,adder_long_double_result,
      divider_long_double_a,divider_long_double_b,divider_long_double_run,
      divider_long_double_ready,divider_long_double_result,substractor_long_double_a,
      substractor_long_double_b,substractor_long_double_run,substractor_long_double_ready,
      substractor_long_double_result,convertor_from_long_double_to_signed_16_operand,
      convertor_from_long_double_to_signed_16_run,convertor_from_long_double_to_signed_16_ready,con
      vertor_from_long_double_to_signed_16_result, result_func,param_func_a,param_func_b,
      param_func_c,param_func_i,run_func,ready_func);

   input          clock, reset, zero_substractor_float_ready, convertor_from_float_to_long_double_ready,
                  convertor_from_double_to_long_double_ready, multiplier_long_double_ready,
                  adder_long_double_ready, divider_long_double_ready, substractor_long_double_ready,
                  convertor_from_long_double_to_signed_16_ready, run_func;
   input  [15:0]  convertor_from_long_double_to_signed_16_result, param_func_i;
   input  [31:0]  zero_substractor_float_result, param_func_a;
   input  [63:0]  param_func_b;
   input  [79:0]  convertor_from_float_to_long_double_result, param_func_c,
                  convertor_from_double_to_long_double_result, multiplier_long_double_ready,
                  adder_long_double_result, divider_long_double_result, substractor_long_double_result;
   output         zero_substractor_float_run, convertor_from_float_to_long_double_run,
                  convertor_from_double_to_long_double_run, multiplier_long_double_run,
                  adder_long_double_run, divider_long_double_run, substractor_long_double_run,
                  convertor_from_long_double_to_signed_16_run, ready_func;
   output [15:0]  result_func;
   output [31:0]  zero_substractor_float_operand, convertor_from_float_to_long_double_operand;
   output [63:0]  convertor_from_double_to_long_double_operand;
   output [79:0]  multiplier_long_double_a, multiplier_long_double_b, adder_long_double_a,
                  adder_long_double_b, divider_long_double_a, divider_long_double_b,
                  substractor_long_double_a, substractor_long_double_b,
                  convertor_from_long_double_to_signed_16_operand;
```

FIG._9B1

```
reg             zero_substractor_float_run, convertor_from_float_to_long_double_run,
                convertor_from_double_to_long_double_run, multiplier_long_double_run,
                adder_long_double_run, divider_long_double_run, substractor_long_double_run,
                convertor_from_long_double_to_signed_16_run, ready_func;
reg     [ 4:0]  state;
reg     [15:0]  result_func, _8_16;
reg     [31:0]  zero_substractor_float_operand, convertor_from_float_to_long_double_operand, _1_32;
reg     [63:0]  convertor_from_double_to_long_double_operand;
reg     [79:0]  multiplier_long_double_a, multiplier_long_double_b, adder_long_double_a,
                adder_long_double_b, divider_long_double_a, divider_long_double_b,
                substractor_long_double_a, substractor_long_double_b,
                convertor_from_long_double_to_signed_16_operand, 2_80, _3_80, _4_80, _5_80,
                _6_80, _7_80;

always @(posedge clock)
  begin
    if (reset)
      state = 0;
    else
    begin
      case (state)
      0: begin
        if (run_func)
        begin
         ready_func = 0;
         zero_substractor_float_operand = param_func_a;
         zero_substractor_float_run = 1;
          state = 1;
        end
      end
      1: begin
        zero_substractor_float_run = 0;
        _1_32 = zero_substractor_float_result;
        if (zero_substractor_float_ready)
          state = 2;
        end
      2: begin
        convertor_from_float_to_long_double_operand = _1_32;
        convertor_from_float_to_long_double_run = 1;
        state = 3;
      end
      3: begin
        convertor_from_float_to_long_double_run = 0;
        _2_80 = convertor_from_float_to_long_double_result;
        if (convertor_from_float_to_long_double_ready)
          state = 4;
      end
```

FIG._9B2

```
4: begin
  convertor_from_double_to_long_double_operand = param_func_b;
  convertor_from_double_to_long_double_run = 1;
  state = 5;
end
5: begin
  convertor_from_double_to_long_double_run = 0;
  _3_80 = convertor_from_double_to_long_double_result;
  if (convertor_from_double_to_long_double_ready)
    state = 6;
end
6: begin
  multiplier_long_double_a = _3_80;
  multiplier_long_double_b = param_func_c;
  multiplier_long_double_run = 1;
  state = 7;
end
7: begin
  multiplier_long_double_run = 0;
  _4_80 = multiplier_long_double_result;
  if (multiplier_long_double_ready)
    state = 8;
end
8: begin
  adder_long_double_a = _2_80;
  adder_long_double_b = _4_80;
  adder_long_double_run = 1;
  state = 9;
end
9: begin
  adder_long_double_run = 0;
  _5_80 = adder_long_double_result;
  if (adder_long_double_ready)
    state = 10;
end
10: begin
  divider_long_double_a = 80'h0000000000803040;
  divider_long_double_b = param_func_c;
  divider_long_double_run = 1;
  state = 11;
end
11: begin
  divider_long_double_run = 0;
  _6_80 = divider_long_double_result;
  if (divider_long_double_ready)
    state = 12;
end
```

FIG._9B3

```
            12: begin
              substractor_long_double_a = _5_80;
              substractor_long_double_b = _6_80;
              substractor_long_double_run = 1;
              state = 13;
            end
            13: begin
              substractor_long_double_run = 0;
              _7_80 = substractor_long_double_result;
              if (substractor_long_double_ready)
                state = 14;
            end
            14: begin
              convertor_from_long_double_to_signed_16_operand = _7_80;
              convertor_from_long_double_to_signed_16_run = 1;
              state = 15;
            end
            15: begin
              convertor_from_long_double_to_signed_16_run = 0;
              _8_16 = convertor_from_long_double_to_signed_16_result;
              if (convertor_from_long_double_to_signed_16_ready)
              state = 16;
            end
            16: begin
              result_func = _8_16;
              ready_func = 1;
              state = 0;
            end
            default: ;
            endcase
          end
        end

endmodule
```

FIG._9B4

```
typedef struct
{
   int real;
   int image;
}
Complex;

Complex add (Complex a, Complex b)
{
   Complex c;
   c.real  = a.real  + b.real;
   c.image = a.image + b.image;
   return c;
}

Complex x = { 1, 2 }, y = { 3, 4 }, z;

void main ()
{
   z = add (x, add (x, y));
}
```

FIG._10A

```
`define _add_a   4'h1
`define _add_b   4'h3
`define _3_add_c 4'h5
`define _x       4'h7
`define _y       4'h9
`define _z       4'hB

module Complex (clock, reset, a, d, o, we, result_add, param_add_a,
                param_add_b, run_add, ready_add, run_main, ready_main);

  input          clock, reset, run_add, run_main;
  input  [15:0]  o;
  input  [31:0]  param_add_a, param_add_b;

  output         we, ready_add, ready_main;
  output [ 3:0]  a;
  output [15:0]  d;
  output [31:0]  result_add;

  reg            we, ready_add, ready_main;
  reg    [ 3:0]  a;
  reg    [ 4:0]  state, return_state_add;
  reg    [15:0]  d, _1_16, _2_16, _3_16, _4_16, _5_16, _6_16;
  reg    [31:0]  result_add, _7_32, _8_32, _9_32, _10_32;

  parameter _add = 1;

  always @(posedge clock)
  begin
    if (reset)
    begin
      we = 0;
      state = 0;
    end
    else
    begin
      case (state)
       0: begin
            if (run_add)
             begin
               ready_add = 0;
               return_state_add = 0;
               a = `_add_a;
               d = param_add_a [31:16];
                we = 1;
                state = 2;
             end
            else if (run_main)
             begin
               ready_main = 0;
               a = `_x;
               state = 14;
             end
         end
```

FIG._10B1

```
2: begin
     we = 0;
    a = `_add_a + 4'd1;
    d = param_add_a [15:0];
     we = 1;
     state = 3;
  end
3: begin
     we = 0;
    a = `_add_b;
    d = param_add_b [31:16];
     we = 1;
     state = 4;
  end
4: begin
     we = 0;
    a = `_add_b + 4'd1;
    d = param_add_b [15:0];
     we = 1;
     state = 5;
  end
5: begin
     we = 0;
    state = _add;
  end
_add: begin
     a = `_add_a;
     state = 6;
  end
6: begin
     _1_16 = o;
    a = `_add_b;
     state = 7;
  end
7: begin
     _2_16 = o;
     _3_16 = (_1_16 + _2_16);
    a = `_3_add_c;
     d = _3_16;
     we = 1;
     state = 8;
  end
8: begin
     we = 0;
    a = (`_add_a + 16'd1);
     state = 9;
  end
```

FIG._10B2

```
9: begin
      _4_16 = o;
     a = (`_add_b + 16'd1);
      state = 10;
   end
10: begin
      _5_16 = o;
      _6_16 = (_4_16 + _5_16);
     a = (`_3_add_c + 16'd1);
      d = _6_16;
      we = 1;
      state = 11;
   end
11: begin
      we = 0;
     a = `_3_add_c;
      state = 12;
   end
12: begin
      _7_32 [31:16] = o;
     a = `_3_add_c + 4'd1;
      state = 13;
   end
13: begin
      _7_32 [15:0] = o;
     result_add = _7_32;
     ready_add = 1;
     state = return_state_add;
   end
14: begin
      _8_32 [31:16] = o;
      a = `_x + 4'd1;
      state = 15;
   end
15: begin
      _8_32 [15:0] = o;
     a = `_add_a;
      d = _8_32 [31:16];
      we = 1;
      state = 16;
   end
16: begin
      we = 0;
     a = `_add_a + 4'd1;
      d = _8_32 [15:0];
      we = 1;
      state = 17;
   end
```

FIG._10B3

```
17: begin
      we = 0;
      a = `_x;
      state = 18;
    end
18: begin
      _9_32 [31:16] = o;
      a = `_x + 4'd1;
      state = 19;
    end
19: begin
      _9_32 [15:0] = o;
      a = `_add_a;
      d = _9_32 [31:16];
      we = 1;
      state = 20;
    end
20: begin
      we = 0;
      a = `_add_a + 4'd1;
      d = _9_32 [15:0];
      we = 1;
      state = 21;
    end
21: begin
      we = 0;
      a = `_y;
      state = 22;
    end
22: begin
      _10_32 [31:16] = o;
      a = `_y + 4'd1;
      state = 23;
    end
23: begin
      _10_32 [15:0] = o;
      a = `_add_b;
      d = _10_32 [31:16];
      we = 1;
      state = 24;
    end
24: begin
      we = 0;
      a = `_add_b + 4'd1;
      d = _10_32 [15:0];
      we = 1;
      state = 25;
    end
```

FIG._10B4

```
25: begin
       we = 0;
      return_state_add = 26;
      state = _add;
   end
26: begin
      a = `_add_b;
      d = result_add [31:16];
       we = 1;
       state = 27;
   end
27: begin
       we = 0;
      a = `_add_b + 4'd1;
      d = result_add [15:0];
       we = 1;
       state = 28;
   end
28: begin
       we = 0;
      return_state_add = 29;
      state = _add;
   end
29: begin
       a = `_z;
      d = result_add [31:16];
       we = 1;
       state = 30;
   end
30: begin
       we = 0;
       a = `_z + 4'd1;
      d = result_add [15:0];
       we = 1;
       state = 31;
   end
31: begin
       we = 0;
      ready_main = 1;
       state = 0;
   end
default: ;
      endcase
   end
 end
endmodule
```

FIG._10B5

```
#ifndef __SYNETRY__
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#else
#define NULL ((NODE *) 0)
#define assert(a)
#endif
#define MAX_NODES 10
struct _Node
{   struct _Node * pLeft;
    struct _Node * pRight;
    int            nKey;
    int            nValue;
};
typedef struct _Node NODE;
static NODE Nodes [MAX_NODES];
static int  nNodes;
NODE * pTree;
void Initialize (void)
{   nNodes = 0;
    pTree  = NULL;
}
static NODE * NewNode (int nKey, int nValue)
{   assert (nNodes >= 0 && nNodes <= MAX_NODES);
    if (nNodes == MAX_NODES)
        return NULL;
    Nodes [nNodes].pLeft  = NULL;
    Nodes [nNodes].pRight = NULL;
    Nodes [nNodes].nKey   = nKey;
    Nodes [nNodes].nValue = nValue;
    return & Nodes [nNodes++];
}

NODE * FindNode (int nKey)
{   NODE * p;
    p = pTree;
    while (p != NULL)
    {
        if (p -> nKey < nKey)
            p = p -> pLeft;
        else if (p -> nKey > nKey)
            p = p -> pRight;
        else
            return p;
    }
    return NULL;
}
int FindValue (int nKey)
{   NODE * p;
    return (p = FindNode (nKey)) == NULL ? -1 : p -> nValue;
}
```

FIG._11A1

```
NODE * FindOrAddNode (nKey, nValue)
int nKey;
int nValue;
{  NODE * * p;
   p = & pTree;
   while (*p != NULL)
   {      if ((*p) -> nKey < nKey)
             p = & (*p) -> pLeft;
       else if ((*p) -> nKey > nKey)
             p = & (*p) -> pRight;
       else
             return (*p);
   }
   return *p = NewNode (nKey, nValue);
}
int FindOrAddValue (nKey, nValue)
int nKey;
int nValue;
{  NODE * p = FindOrAddNode (nKey, nValue);
   return p == NULL ? -1 : p -> nValue;
}
#ifndef __SYNETRY__
void PrintTree (int nLevel, NODE * p)
{  if (p == NULL)
       return;
   PrintTree (nLevel + 1, p -> pLeft);
   printf ("%*s%d [%d]\n", nLevel * 4, "", p -> nKey, p -> nValue);
   PrintTree (nLevel + 1, p -> pRight);
}

void main ()
{  int anKeys [20]
       = { 12, 4, 7, 2, 18, 24, 0, 5, 14, 1, 43, 6, 19, 21, 26, 11, 37, 8, 4, 9};
   int i;

   Initialize ();
   for (i = 0; i < 20; i++)
      FindOrAddValue (anKeys [i], i);
   for (i = 0; i < 20; i++)
      printf ("%d: %d\n", i, FindValue (i));
   printf ("\nTree:\n");
   PrintTree (0, pTree);
}
#endif
```

FIG._11A2

```
`define _Nodes 6'h1
`define _pTree 6'h29

module Tree ( clock, reset, a, d, o, we, result_FindNode, result_FindValue, result_FindOrAddNode,
              result_FindOrAddValue, param_FindNode_nKey, param_FindValue_nKey,
              param_FindOrAddNode_nKey, param_FindOrAddNode_nValue,
              param_FindOrAddValue_nKey, param_FindOrAddValue_nValue,
              run_Initialize, ready_Initialize, run_FindNode, ready_FindNode,
              run_FindValue, ready_FindValue, run_FindOrAddNode,
              ready_FindOrAddNode, run_FindOrAddValue, ready_FindOrAddValue );
   input        clock, reset, run_Initialize, run_FindNode, run_FindValue, run_FindOrAddNode,
                run_FindOrAddValue;
   input  [15:0] o, param_FindNode_nKey, param_FindValue_nKey, param_FindOrAddNode_nKey,
                param_FindOrAddNode_nValue, param_FindOrAddValue_nKey,
                param_FindOrAddValue_nValue;
   output       we, ready_Initialize, ready_FindNode, ready_FindValue, ready_FindOrAddNode,
                ready_FindOrAddValue;
   output [ 5:0] a, result_FindNode, result_FindOrAddNode, d, result_FindValue, result_FindOrAddValue;
   reg          we, ready_Initialize, ready_FindNode, ready_FindValue, ready_FindOrAddNode,
                ready_FindOrAddValue;
   reg    [ 5:0] a, result_FindNode, result_FindOrAddNode, d, result_FindValue, result_FindOrAddValue,
                state, result_NewNode, _7_FindNode_p, _10_FindValue_p, _13_FindOrAddNode_p,
                _17_FindOrAddValue_p, return_state_NewNode, return_state_FindNode,
                return_state_FindOrAddNode, _1_6, _2_6, _3_6, _4_6, _5_6, _6_6, _7_6, _8_6, _9_6,
                _10_6, _12_6, _13_6, _14_6, _18_6, _22_6, _23_6, _24_6, _27_6, _28_6, _29_6,
                _33_6, _34_6, _38_6, _39_6, _40_6;
   reg    [15:0] _nNodes, _NewNode_nKey, _NewNode_nValue, _FindNode_nKey,
                _FindOrAddNode_nKey, _FindOrAddNode_nValue, _30_16, _35_16, _11_16, _15_16,
                _19_16, _25_16, _26_16, _41_16, _42_16;
   reg    [16:0] _16_17, _17_17, _20_17, _21_17, _31_17, _32_17, _36_17, _37_17;

   parameter    _NewNode      = 1,
                _FindNode     = 5,
                _FindOrAddNode = 14;

   always @(posedge clock)
   begin
      if (reset)
      begin
         we = 0;
         state = 0;
      end
```

FIG._11B1

```
else
begin
   case (state)
    0: begin
       if (run_Initialize)
       begin
          ready_Initialize = 0;
          _nNodes = 16'd0;
          _1_6 = 16'd0;
          a = `_pTree;
          d [15:10] = _1_6;
          we = 1;
          state = 23;
       end
       else if (run_FindNode)
       begin
          ready_FindNode = 0;
          return_state_FindNode = 0;
          _FindNode_nKey = param_FindNode_nKey;
          state = _FindNode;
       end
       else if (run_FindValue)
       begin
          ready_FindValue = 0;
          _FindNode_nKey = param_FindValue_nKey;
          return_state_FindNode = 33;
          state = _FindNode;
       end
       else if (run_FindOrAddNode)
       begin
          ready_FindOrAddNode = 0;
          return_state_FindOrAddNode = 0;
          _FindOrAddNode_nKey = param_FindOrAddNode_nKey;
          _FindOrAddNode_nValue = param_FindOrAddNode_nValue;
          state = _FindOrAddNode;
       end
       else if (run_FindOrAddValue)
       begin
          ready_FindOrAddValue = 0;
          _FindOrAddNode_nKey = param_FindOrAddValue_nKey;
          _FindOrAddNode_nValue = param_FindOrAddValue_nValue;
          return_state_FindOrAddNode = 48;
          state = _FindOrAddNode;
       end
       end

       23: begin
          we = 0;
          ready_Initialize = 1;
          state = 0;
       end

   _NewNode:  state = (_nNodes == 16'd10) ? 2 : 3;
```

FIG._11B2

```
2: begin
   _2_6 = 16'd0;
   result_NewNode = _2_6;
   state = 4;
end

3: begin
   _3_6 = 16'd0;
   _4_6 = (_nNodes << 2'd2);
   _5_6 = _3_6;
   a = (`_Nodes + _4_6);
   d [15:10] = _5_6;
   we = 1;
   state = 24;
end

24: begin
   we = 0;
   _6_6 = 16'd0;
   _7_6 = (_nNodes << 2'd2);
   _8_6 = _6_6;
   a = ((`_Nodes + _7_6) + 6'd1);
   d [15:10] = _8_6;
   we = 1;
   state = 25;
end

25: begin
   we = 0;
   _9_6 = (_nNodes << 2'd2);
   a = ((`_Nodes + _9_6) + 6'd2);
   d = _NewNode_nKey;
   we = 1;
   state = 26;
end

26: begin
   we = 0;
   _10_6 = (_nNodes << 2'd2);
   a = ((`_Nodes + _10_6) + 6'd3);
   d = _NewNode_nValue;
   we = 1;
   state = 27;
end

27: begin
   we = 0;
   _11_16 = _nNodes;
   _nNodes = (_11_16 + 16'd1);
   _12_6 = (_11_16 << 2'd2);
   result_NewNode = (`_Nodes + _12_6);
   state = 4;
end
```

FIG._11B3

```
4: state = return_state_NewNode;

FindNode: begin
    a = `_pTree;
    state = 28;
end

28: begin
    _13_6 = o [15:10];
    _7_FindNode_p = _13_6;
    state = 6;
end

6: begin
    _14_6 = 16'd0;
    state = (_7_FindNode_p != _14_6) ? 7 : 12;
end

7: begin
    a = (_7_FindNode_p + 6'd2);
    state = 29;
end

29: begin
    _15_16 = o;
    _16_17 = { ~ _15_16 [15], _15_16 };
    _17_17 = { ~ _FindNode_nKey [15], _FindNode_nKey };
    state = (_16_17 < _17_17) ? 8 : 9;
end

8: begin
    a = _7_FindNode_p;
    state = 30;
end

30: begin
    _18_6 = o [15:10];
    _7_FindNode_p = _18_6;
    state = 6;
end

9: begin
    a = (_7_FindNode_p + 6'd2);
    state = 31;
end

31: begin
    _19_16 = o;
    _20_17 = { ~ _19_16 [15], _19_16 };
    _21_17 = { ~ _FindNode_nKey [15], _FindNode_nKey };
    state = (_20_17 > _21_17) ? 10 : 11;
end

10: begin
    a = (_7_FindNode_p + 6'd1);
    state = 32;
end
```

FIG._11B4

```
32: begin
  _22_6 = o [15:10];
  _7_FindNode_p = _22_6;
  state = 6;
end
11: begin
  result_FindNode = _7_FindNode_p;
  state = 13;
end
12: begin
  _23_6 = 16'd0;
  result_FindNode = _23_6;
  state = 13;
end
13: begin
  ready_FindNode = 1;
  state = return_state_FindNode;
end
33: begin
  _10_FindValue_p = result_FindNode;
  _24_6 = 16'd0;
  state = (result_FindNode == _24_6) ? 34 : 35;
end
34: begin
  _25_16 = -16'd1;
  state = 36;
end
35: begin
  a = (_10_FindValue_p + 6'd3);
  state = 37;
end
37: begin
  _26_16 = o;
  _25_16 = _26_16;
  state = 36;
end
36: begin
  result_FindValue = _25_16;
  ready_FindValue = 1;
  state = 0;
end
```

FIG._11B5

```
_FindOrAddNode: begin
    _13_FindOrAddNode_p = `_pTree;
    state = 15;
end

15: begin
    a = _13_FindOrAddNode_p;
    state = 38;
end

38: begin
    _27_6 = o [15:10];
    _28_6 = 16'd0;
    state = (_27_6 != _28_6) ? 16 : 21;
end

16: begin
    a = _13_FindOrAddNode_p;
    state = 39;
end

39: begin
    _29_6 = o [15:10];
    a = (_29_6 + 6'd2);
    state = 40;
end

40: begin
    _30_16 = o;
    _31_17 = { ~ _30_16 [15], _30_16 };
    _32_17 = { ~ _FindOrAddNode_nKey [15], _FindOrAddNode_nKey };
    state = (_31_17 < _32_17) ? 17 : 18;
end

17: begin
    a = _13_FindOrAddNode_p;
    state = 41;
end

41: begin
    _33_6 = o [15:10];
    _13_FindOrAddNode_p = _33_6;
    state = 15;
end

18: begin
    a = _13_FindOrAddNode_p;
    state = 42;
end

42: begin
    _34_6 = o [15:10];
    a = (_34_6 + 6'd2);
    state = 43;
end
```

FIG._11B6

```
43: begin
    _35_16 = o;
    _36_17 = { ~ _35_16 [15], _35_16 };
  _37_17 = { ~ _FindOrAddNode_nKey [15], _FindOrAddNode_nKey };
    state = (_36_17 > _37_17) ? 19 : 20;
end

19: begin
   a = _13_FindOrAddNode_p;
    state = 44;
end

44: begin
    _38_6 = o [15:10];
  _13_FindOrAddNode_p = (_38_6 + 6'd1);
    state = 15;
end

20: begin
   a = _13_FindOrAddNode_p;
    state = 45;
end

45: begin
    _39_6 = o [15:10];
  result_FindOrAddNode = _39_6;
    state = 22;
end

21: begin
  _NewNode_nKey = _FindOrAddNode_nKey;
  _NewNode_nValue = _FindOrAddNode_nValue;
   return_state_NewNode = 46;
    state = _NewNode;
end

46: begin
   a = _13_FindOrAddNode_p;
   d [15:10] = result_NewNode;
    we = 1;
    state = 47;
end

47: begin
    we = 0;
  result_FindOrAddNode = result_NewNode;
    state = 22;
end
```

FIG._11B7

```
22: begin
  ready_FindOrAddNode = 1;
  state = return_state_FindOrAddNode;
end

48: begin
  _17_FindOrAddValue_p = result_FindOrAddNode;
    _40_6 = 16'd0;
  state = (_17_FindOrAddValue_p == _40_6) ? 49 : 50;
end

49: begin
    _41_16 = -16'd1;
    state = 51;
end

50: begin
   a = (_17_FindOrAddValue_p + 6'd3);
    state = 52;
end

52: begin
    _42_16 = o;
    _41_16 = _42_16;
    state = 51;
end

51: begin
   result_FindOrAddValue = _41_16;
   ready_FindOrAddValue = 1;
    state = 0;
end

  default: ;
  endcase
  end
end

endmodule
```

FIG._11B8

```
#include <stdio.h>
int event_occured = 0;
#pragma synchronous on_event

void on_event ()
{
  event_occured = 1;
}

void main ()
{
   int a, b, c;
   scanf ("%d %d", & a, & b);
   c = event_occured ? a + b : a - b;
   printf ("%d\n", c);
}
```

FIG._12A

```
module Event_sy (clock,reset,scanf_0_1_line_16_a,scanf_0_2_line_16_b,printf_1_1_line_20_c,
                        run_on_event,run_main);
   input          clock, reset, run_on_event, run_main;
   input  [15:0]  scanf_0_1_line_16_a, scanf_0_2_line_16_b;
   output [15:0]  printf_1_1_line_20_c;
   reg    [15:0]  printf_1_1_line_20_c, _event_occured, _108_main_a, _108_main_b, _108_main_c;

   always @(posedge clock)
   begin
      if (run_on_event)
         _event_occured = 16'd1;
      if (reset)
         _event_occured = 16'd0;
   end

   always @(posedge clock)
   begin
      if (run_main)
      begin
         _108_main_a = scanf_0_1_line_16_a;
         _108_main_b = scanf_0_2_line_16_b;
         _108_main_c = (_event_occured ? (_108_main_a + _108_main_b) :
                              (_108_main_a - _108_main_b));
         printf_1_1_line_20_c = _108_main_c;
      end
   end

endmodule
```

FIG._12B

```
#include <stdio.h>

int event_occured = 0;
#pragma asynchronous on_event
void on_event ()
{
  event_occured = 1;
}

void main ()
{
   int a, b, c;
   scanf ("%d %d", & a, & b);
   c = event_occured ? a + b : a - b;
   printf ("%d\n", c);
}
```

FIG._13A

```
module Event_as(clock,reset,scanf_0_1_line_16_a,scanf_0_2_line_16_b,printf_1_1_line_20_c,
                                run_on_event,run_main);
   input          clock, reset, run_on_event, run_main;
   input  [15:0]  scanf_0_1_line_16_a, scanf_0_2_line_16_b;
   output [15:0]  printf_1_1_line_20_c;
   reg    [15:0]  printf_1_1_line_20_c, _event_occured, _108_main_a, _108_main_b, _108_main_c;

  always @(posedge run_on_event)
   begin
      if (reset)
        _event_occured = 16'd0;
      else
        _event_occured = 16'd1;
   end

  always @(posedge clock)
   begin
      if (run_main)
      begin
        _108_main_a = scanf_0_1_line_16_a;
        _108_main_b = scanf_0_2_line_16_b;
        _108_main_c = (_event_occured ? (_108_main_a + _108_main_b) :
                                        (_108_main_a - _108_main_b));
        printf_1_1_line_20_c = _108_main_c;
      end
   end

endmodule
```

FIG._13B

```
int func1 (unsigned n)
{
   unsigned i, result = 0;
   for (i = 0; i <= n; i++)
      result += i;
   return result;
}

int func2 (unsigned n)
{
   unsigned i, result = 0;
   for (i = 0; i <= n; i++)
      result += i;
   return result;
}
```

FIG._14A

```
module Sum (clock, reset, result__func1, param__func1__n, run__func1, ready__func1, result__func2,
                  param__func2__n, run__func2, ready__func2 );
  input          clock, reset, run__func1, run__func2;
  input  [15:0]  param__func1__n, param__func2__n;
  output         ready__func1, ready__func2;
  output [15:0]  result__func1, result__func2;
  reg            ready__func1, ready__func2;
  reg     [1:0]  state1, state2;
  reg    [15:0]  result__func1, result__func2, s_2_func1_i, s_2_func1_result, s_2_func2_i,
                  s_2_func2_result;;
  always @(posedge clock)
     begin
        if (reset)
           state1 = 0;
        else
        begin
           case (state1)
           0: begin
              if (run__func1)
              begin
                 ready__func1 = 0;
                 s_2_func1_result = 16'd0;
                 s_2_func1_i = 16'd0;
                 state1 = 1;
              end
           end
```

FIG._14B1

```
1: state1 = (s_2_func1_i <= param__func1__n) ? 2 : 3;

2: begin
   s_2_func1_result = (s_2_func1_result + s_2_func1_i);
   s_2_func1_i = (s_2_func1_i + 16'd1);
   state1 = 1;
end

3: begin
   result__func1 = s_2_func1_result;
   ready__func1 = 1;
   state1 = 0;
end
endcase
end
end

always @(posedge clock)
    begin
      if (reset)
        state2 = 0;
      else
      begin
        case (state2)
        0: begin
          if (run__func2)
          begin
            ready__func2 = 0;
            s_2_func2_result = 16'd0;
            s_2_func2_i = 16'd0;
            state2 = 1;
          end
        end

        1: state2 = (s_2_func2_i <= param__func2__n) ? 2 : 3;

        2: begin
          s_2_func2_result = (s_2_func2_result + s_2_func2_i);
          s_2_func2_i = (s_2_func2_i + 16'd1);
          state2 = 1;
        end

        3: begin
          result__func2 = s_2_func2_result;
          ready__func2 = 1;
          state2 = 0;
        end
        endcase
      end
    end

  endmodule
```

FIG._14B2

```
#include <stdio.h>
#include <stdarg.h>
/* Returns the average of a variable list of integers. */

static int average (int first, ...)
{
  int count = 0, sum = 0, i = first;
  va_list marker;

  va_start (marker, first);    /* Initialize variable arguments. */

  while (i != -1)
  {
    sum += i;
    count++;
    i = va_arg (marker, int);
  }

  va_end (marker);             /* Reset variable arguments.      */

  return sum ? (sum / count) : 0;
}

void main ()
{
  /* Call with 3 integers (-1 is used as terminator). */
  printf ("Average is: %d\n", average (2, 3, 4, -1));

  /* Call with 4 integers. */
  printf ("Average is: %d\n", average (5, 7, 9, 11, -1));

  /* Call with just -1 terminator. */
  printf ("Average is: %d\n", average (-1));
}
```

FIG._15A

```
`define _average_first 5'h1
module udivmod16 (a, b, div, mod);
   input  [15:0]  a, b;
   output [15:0]  div, mod;
   reg    [15:0]  div, mod, pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7, pa8, pa9, pa10,
                  pa11, pa12, pa13, pa14, pa15, pa16, shb0;
   reg    [16:0]  shb1;
   reg    [17:0]  shb2;
   reg    [18:0]  shb3;
   reg    [19:0]  shb4;
   reg    [20:0]  shb5;
   reg    [21:0]  shb6;
   reg    [22:0]  shb7;
   reg    [23:0]  shb8;
   reg    [24:0]  shb9;
   reg    [25:0]  shb10;
   reg    [26:0]  shb11;
   reg    [27:0]  shb12;
   reg    [28:0]  shb13;
   reg    [29:0]  shb14;
   reg    [30:0]  shb15;

   always @(a or b)
   begin
      pa0 = a;

      shb15    = b << 15;
      div [15] = pa0 >= shb15;
      pa1      = div [15] ? pa0 - shb15 : pa0;

      shb14    = b << 14;
      div [14] = pa1 >= shb14;
      pa2      = div [14] ? pa1 - shb14 : pa1;

      shb13    = b << 13;
      div [13] = pa2 >= shb13;
      pa3      = div [13] ? pa2 - shb13 : pa2;

      shb12    = b << 12;
      div [12] = pa3 >= shb12;
      pa4      = div [12] ? pa3 - shb12 : pa3;

      shb11    = b << 11;
      div [11] = pa4 >= shb11;
      pa5      = div [11] ? pa4 - shb11 : pa4;

      shb10    = b << 10;
      div [10] = pa5 >= shb10;
      pa6      = div [10] ? pa5 - shb10 : pa5;

      shb9     = b << 9;
      div [ 9] = pa6 >= shb9;
      pa7      = div [9] ? pa6 - shb9 : pa6;
```

FIG._15B1

```
      shb8     = b << 8;
      div [ 8] = pa7 >= shb8;
      pa8      = div [8] ? pa7 - shb8 : pa7;

      shb7     = b << 7;
      div [ 7] = pa8 >= shb7;
      pa9      = div [7] ? pa8 - shb7 : pa8;

      shb6     = b << 6;
      div [ 6] = pa9 >= shb6;
      pa10     = div [6] ? pa9 - shb6 : pa9;

      shb5     = b << 5;
      div [ 5] = pa10 >= shb5;
      pa11     = div [5] ? pa10 - shb5 : pa10;

      shb4     = b << 4;
      div [ 4] = pa11 >= shb4;
      pa12     = div [4] ? pa11 - shb4 : pa11;

      shb3     = b << 3;
      div [ 3] = pa12 >= shb3;
      pa13     = div [3] ? pa12 - shb3 : pa12;

      shb2     = b << 2;
      div [ 2] = pa13 >= shb2;
      pa14     = div [2] ? pa13 - shb2 : pa13;

      shb1     = b << 1;
      div [ 1] = pa14 >= shb1;
      pa15     = div [1] ? pa14 - shb1 : pa14;

      shb0     = b << 0;
      div [ 0] = pa15 >= shb0;
      pa16     = div [0] ? pa15 - shb0 : pa15;

      mod = pa16;
   end
endmodule

module sdivmod16 (a, b, div, mod);
   input  [15:0]  a, b;
   output [15:0]  div, mod;
   wire   [15:0]  ua  =  a [15] ? - a : a;
   wire   [15:0]  ub  =  b [15] ? - b : b;
   wire   [15:0]  udiv, umod;
   udivmod16 udivmod (ua, ub, udiv, umod);
   wire   [15:0]  div = (a [15] != b [15]) ? - udiv : udiv;
   wire   [15:0]  mod = a [15]  ? - umod : umod;
endmodule
```

FIG._15B2

```
module RAM32X1 (O, D, WE, A4, A3, A2, A1, A0)
  /* synthesis black_box xc_alias="RAM" */ ;
  output        O;
  input         D, WE, A4, A3, A2, A1, A0;
  reg [31:0]    d;
  wire [4:0]    A = { A4, A3, A2, A1, A0 };
  assign O = d [A];
  always @(A or D or WE)
     if (WE)
        d [A] = D;
endmodule

module RAM32x1 (a, d, o, we);
  input                 d, we;
  input [4:0]     a;
  output          o;
  RAM32X1 RAM32X1 (o, d, we, a [4], a [3], a [2], a [1], a [0]);
endmodule

module RAM32x16 (a, d, o, we);
  input          we;
  input  [ 4:0]  a;
  input  [15:0]  d;
  output [15:0]  o;
  RAM32x1 U0  (a, d [ 0], o [ 0], we);
  RAM32x1 U1  (a, d [ 1], o [ 1], we);
  RAM32x1 U2  (a, d [ 2], o [ 2], we);
  RAM32x1 U3  (a, d [ 3], o [ 3], we);
  RAM32x1 U4  (a, d [ 4], o [ 4], we);
  RAM32x1 U5  (a, d [ 5], o [ 5], we);
  RAM32x1 U6  (a, d [ 6], o [ 6], we);
  RAM32x1 U7  (a, d [ 7], o [ 7], we);
  RAM32x1 U8  (a, d [ 8], o [ 8], we);
  RAM32x1 U9  (a, d [ 9], o [ 9], we);
  RAM32x1 U10 (a, d [10], o [10], we);
  RAM32x1 U11 (a, d [11], o [11], we);
  RAM32x1 U12 (a, d [12], o [12], we);
  RAM32x1 U13 (a, d [13], o [13], we);
  RAM32x1 U14 (a, d [14], o [14], we);
  RAM32x1 U15 (a, d [15], o [15], we);
endmodule

module RAM18x16 (a, d, o, we);
  input          we;
  input  [ 4:0]  a;
  input  [15:0]  d;
  output [15:0]  o;
  RAM32x16 U (a, d, o, we);
endmodule
```

FIG._15B3

```
module StdArg (clock, reset, printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48,
                run_main, ready_main );
  input         clock, reset, run_main;
  output        ready_main;
  output [15:0] printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48
  reg           ready_main, we;
  reg    [ 4:0] state, a, return_state_average;
  reg    [15:0] printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48, divmod_a, divmod_b,
                d, result_average, _108_average_count, _108_average_sum, _108_average_i,
                _108_average_marker, _1_16, _2_16, _3_16, _4_16, _5_16, _6_16, _7_16, _8_16,
                _9_16, _10_16, _11_16, _12_16, _13_16, _14_16;
  wire   [15:0] div_result, o;
  sdivmod16 sdivmod (divmod_a, divmod_b, div_result, );

  RAM18x16 m (a, d, o, we);
  parameter _average = 1;

  always @(posedge clock)
  begin
    if (reset)
    begin
      we = 0;
      state = 0;
    end
    else
    begin
      case (state)

       0: begin
          if (run_main)
          begin
            ready_main = 0;
            _5_16 = 16'd2;
            a = `_average_first;
            d = _5_16;
            we = 1;
            state = 11;
          end
        end

        _average: begin
          _108_average_count = 16'd0;
          _108_average_sum = 16'd0;
          a = `_average_first;
          state = 5;
        end

        5: begin
          _1_16 = o;
          _108_average_i = _1_16;
          _108_average_marker = (`_average_first + 16'd1);
          state = 2;
        end

        2: state = (_108_average_i != -16'd1) ? 3 : 4;
```

FIG._15B4

```
3: begin
 _108_average_sum = (_108_average_sum + _108_average_i);
 _108_average_count = (_108_average_count + 16'd1);
 _108_average_marker = (_108_average_marker + 16'd1);
 a = (_108_average_marker - 16'd1);
  state = 6;
end

6: begin
   _2_16 = o;
  _108_average_i = _2_16;
   state = 2;
end

4: begin
  _108_average_marker = 16'd0;
   state = _108_average_sum ? 7 : 8;
end

7: begin
  divmod_a = _108_average_sum;
  divmod_b = _108_average_count;
   state = 10;
end

10: begin
   _4_16 = div_result;
   _3_16 = _4_16;
   state = 9;
end

8: begin
   _3_16 = 16'd0;
   state = 9;
end

9: begin
  result_average = _3_16;
  state = return_state_average;
end
11: begin
   we = 0;
   _6_16 = 16'd3;
  a = (_average_first + 16'd1);
   d = _6_16;
   we = 1;
   state = 12;
end
```

FIG._15B5

```
12: begin
    we = 0;
    _7_16 = 16'd4;
   a = (`_average_first + 16'd2);
    d = _7_16;
    we = 1;
    state = 13;
end

13: begin
    we = 0;
    _8_16 = -16'd1;
   a = (`_average_first + 16'd3);
    d = _8_16;
    we = 1;
    state = 14;
end

14: begin
    we = 0;
   return_state_average = 15;
   state = _average;
end

15: begin
   printf_1_1_line_42 = result_average;
   // User Verilog code
   $write ("Average is: %d\n", printf_1_1_line_42);
   // End of user Verilog code
    _9_16 = 16'd5;
   a = `_average_first;
    d = _9_16;
    we = 1;
    state = 16;
end

16: begin
    we = 0;
    _10_16 = 16'd7;
   a = (`_average_first + 16'd1);
    d = _10_16;
    we = 1;
    state = 17;
end

17: begin
    we = 0;
    _11_16 = 16'd9;
   a = (`_average_first + 16'd2);
    d = _11_16;
    we = 1;
    state = 18;
end
```

FIG._15B6

```
18: begin
    we = 0;
    _12_16 = 16'd11;
   a = (`_average_first + 16'd3);
    d = _12_16;
    we = 1;
    state = 19;
 end

19: begin
    we = 0;
    _13_16 = -16'd1;
   a = (`_average_first + 16'd4);
    d = _13_16;
    we = 1;
    state = 20;
 end

 20: begin
    we = 0;
   return_state_average = 21;
   state = _average;
 end

 21: begin
   printf_2_1_line_45 = result_average;
   // User Verilog code
   $write ("Average is: %d\n", printf_2_1_line_45);
   // End of user Verilog code
    _14_16 = -16'd1;
   a = `_average_first;
    d = _14_16;
    we = 1;
    state = 22;
 end

 22: begin
    we = 0;
   return_state_average = 23;
   state = _average;
 end

 23: begin
   printf_3_1_line_48 = result_average;
   // User Verilog code
   $write ("Average is: %d\n", printf_3_1_line_48);
   // End of user Verilog code
   ready_main = 1;
    state = 0;
 end

 default: ;
endcase
end
end
endmodule
```

FIG._15B7

```
#include <stdio.h>

static void MoveRing (int nRings, int Tower1, int Tower2, int Tower3)
{
   if (nRings == 0)
      return;

   MoveRing (nRings - 1, Tower1, Tower3, Tower2);

   printf ("%d -> %d\n", Tower1, Tower2);

   MoveRing (nRings - 1, Tower3, Tower2, Tower1);
}

void Hanoi (int nRings)
{
    MoveRing (nRings, 1, 2, 3);
}

#ifndef __SYNETRY__
void main (void)
{
   printf ("Five rings Hanoi Towers\n");
   Hanoi (5);
}
#endif
```

FIG._16A

```
`define recursion_stack 8'h1
module Hanoi (clock, reset, a, d, o, we, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2,
                  param_Hanoi_nRings, run_Hanoi, ready_Hanoi );
   input          clock, reset, run_Hanoi;
   input  [ 7:0]  o, param_Hanoi_nRings;
   output         we, ready_Hanoi;
   output [ 7:0]  a, d, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2;
   reg            we, ready_Hanoi;
   reg    [ 4:0]  state, return_state_MoveRing;
   reg    [ 7:0]  a, d, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2, recursion_stack_pointer,
                  _MoveRing_nRings, _MoveRing_Tower1, _MoveRing_Tower2, _MoveRing_Tower3,
                  _1_8, _2_8, _3_8, _4_8, _5_8, _6_8, _7_8, _8_8, _9_8, _10_8, _11_8, _12_8;
   parameter _MoveRing = 1;

   always @(posedge clock)
   begin
      if (reset)
      begin
         we = 0;
         recursion_stack_pointer = `recursion_stack;
         state = 0;
      end
      else
      begin
         case (state)

         0: begin
            if (run_Hanoi)
            begin
               ready_Hanoi = 0;
               _MoveRing_nRings = param_Hanoi_nRings;
               _MoveRing_Tower1 = 8'd1;
               _MoveRing_Tower2 = 8'd2;
               _MoveRing_Tower3 = 8'd3;
               return_state_MoveRing = 26;
               state = _MoveRing;
            end
         end

         _MoveRing: state = (_MoveRing_nRings == 8'd0) ? 3 : 2;

         2: begin
            _1_8 = (_MoveRing_nRings - 8'd1);
            _2_8 = _MoveRing_Tower1;
            _3_8 = _MoveRing_Tower3;
            _4_8 = _MoveRing_Tower2;
            a = recursion_stack_pointer;
            d = _MoveRing_nRings;
            we = 1;
            state = 4;
         end
```

FIG._16B1

```
4: begin
   we = 0;
   a = recursion_stack_pointer + 1;
   d = _MoveRing_Tower1;
   we = 1;
   state = 5;
end

5: begin
   we = 0;
   a = recursion_stack_pointer + 2;
   d = _MoveRing_Tower2;
   we = 1;
   state = 6;
end

6: begin
   we = 0;
   a = recursion_stack_pointer + 3;
   d = _MoveRing_Tower3;
   we = 1;
   state = 7;
end

7: begin
   we = 0;
   _9_8 = return_state_MoveRing;
   a = recursion_stack_pointer + 4;
   d = _9_8;
   we = 1;
   state = 8;
end

8: begin
   we = 0;
   recursion_stack_pointer = recursion_stack_pointer + 5;
   _MoveRing_nRings = _1_8;
   _MoveRing_Tower1 = _2_8;
   _MoveRing_Tower2 = _3_8;
   _MoveRing_Tower3 = _4_8;
   return_state_MoveRing = 9;
   state = _MoveRing;
end

9: begin
   recursion_stack_pointer = recursion_stack_pointer - 5;
   a = recursion_stack_pointer;
   state = 10;
end

10: begin
   _MoveRing_nRings = o;
   a = recursion_stack_pointer + 1;
   state = 11;
end
```

FIG._16B2

```
11: begin
    _MoveRing_Tower1 = o;
    a = recursion_stack_pointer + 2;
    state = 12;
  end

12: begin
    _MoveRing_Tower2 = o;
    a = recursion_stack_pointer + 3;
    state = 13;
  end

13: begin
    _MoveRing_Tower3 = o;
    a = recursion_stack_pointer + 4;
    state = 14;
  end

14: begin
    _10_8 = o;
    return_state_MoveRing = _10_8;
    printf_1_1_line_22_Tower1 = _MoveRing_Tower1;
    printf_1_2_line_22_Tower2 = _MoveRing_Tower2;
    // User Verilog code
    $write ("%d -> %d\n", printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2);
    // End of user Verilog code
    _5_8 = (_MoveRing_nRings - 8'd1);
    _6_8 = _MoveRing_Tower3;
    _7_8 = _MoveRing_Tower2;
    _8_8 = _MoveRing_Tower1;
    a = recursion_stack_pointer;
    d = _MoveRing_nRings;
    we = 1;
    state = 15;
  end

15: begin
    we = 0;
    a = recursion_stack_pointer + 1;
    d = _MoveRing_Tower1;
    we = 1;
    state = 16;
  end

16: begin
    we = 0;
    a = recursion_stack_pointer + 2;
    d = _MoveRing_Tower2;
    we = 1;
    state = 17;
  end
```

FIG._16B3

```
17: begin
    we = 0;
   a = recursion_stack_pointer + 3;
   d = _MoveRing_Tower3;
    we = 1;
    state = 18;
end
18: begin
    we = 0;
   _11_8 = return_state_MoveRing;
   a = recursion_stack_pointer + 4;
   d = _11_8;
    we = 1;
    state = 19;
end
19: begin
    we = 0;
  recursion_stack_pointer = recursion_stack_pointer + 5;
    _MoveRing_nRings = _5_8;
   _MoveRing_Tower1 = _6_8;
   _MoveRing_Tower2 = _7_8;
   _MoveRing_Tower3 = _8_8;
   return_state_MoveRing = 20;
    state = _MoveRing;
end
20: begin
  recursion_stack_pointer = recursion_stack_pointer - 5;
   a = recursion_stack_pointer;
    state = 21;
end
21: begin
    _MoveRing_nRings = 0;
   a = recursion_stack_pointer + 1;
    state = 22;
end
22: begin
    _MoveRing_Tower1 = 0;
   a = recursion_stack_pointer + 2;
    state = 23;
end
23: begin
    _MoveRing_Tower2 = 0;
   a = recursion_stack_pointer + 3;
    state = 24;
end
```

FIG._16B4

```
        24: begin
            _MoveRing_Tower3 = o;
            a = recursion_stack_pointer + 4;
            state = 25;
        end
        25: begin
            _12_8 = o;
            return_state_MoveRing = _12_8;
            state = 3;
        end
        3: state = return_state_MoveRing;
        26: begin
            ready_Hanoi = 1;
            state = 0;
        end
        default: ;
        endcase
    end
  end
endmodule
```

FIG._16B5

```
#include <stdio.h>

void f1 ()
{
   printf ("%d\n", 1);
}

void f2 ()
{
   printf ("%d\n", 2);
}

void main (n)
{
   void (* f) ();

   f = n == 1 ? f1 : f2;
   f ();
}
```

FIG._17A

```
module PtrFtn (clock, reset, printf_1_1_line_19, printf_2_1_line_24, param_main_n, run_f1, ready_f1,
                run_f2, ready_f2, run_main, ready_main);
   input          clock, reset, run_f1, run_f2, run_main;
   input  [15:0]  param_main_n;
   output         ready_f1, ready_f2, ready_main;
   output [15:0]  printf_1_1_line_19, printf_2_1_line_24;
   reg            ready_f1, ready_f2, ready_main;
   reg    [ 1:0]  state, return_state_f1, return_state_f2;
   reg    [15:0]  printf_1_1_line_19, printf_2_1_line_24, _110_main_f;
  parameter _f1 = 1, _f2 = 2;

   always @(posedge clock)
   begin
      if (reset)
      begin
         printf_1_1_line_19 = 0;
         printf_2_1_line_24 = 0;
         _110_main_f      = 0;
        ready_f1          = 0;
        return_state_f1   = 0;
        ready_f2          = 0;
        return_state_f2   = 0;
        ready_main        = 0;
         state = 0;
      end
```

FIG._17B1

```
else
begin
  case (state)
   0: begin
      if (run_f1)
      begin
        ready_f1 = 0;
        return_state_f1 = 0;
        state = _f1;
      end
      else if (run_f2)
      begin
        ready_f2 = 0;
        return_state_f2 = 0;
        state = _f2;
      end
      else if (run_main)
      begin
        ready_main = 0;
        _110_main_f = ((param_main_n == 16'd1) ? _f1 : _f2);
        state = _110_main_f;
      end
    end
    _f1: begin
        printf_1_1_line_19 = 16'd1;
        // User Verilog code
        $write ("%d\n", printf_1_1_line_19);
        // End of user Verilog code
        ready_f1 = 1;
        state = return_state_f1;
      end
      _f2: begin
        printf_2_1_line_24 = 16'd2;
        // User Verilog code
        $write ("%d\n", printf_2_1_line_24);
        // End of user Verilog code
        ready_f2 = 1;
        state = return_state_f2;
      end
      3: begin
        ready_main = 1;
        state = 0;
      end
      default: ;
      endcase
    end
  end
endmodule
```

FIG._17B2

```
int sum (int array [], int size)
{
    int i, sum = 0;

    for (i = 0; i < size; i ++)
        sum += array [i];

    return sum;
}

int main ()
{
    int i;
    int array [10];
    int size = sizeof (array) / sizeof (*array);

    array [0] = 1;
    array [1] = 2;
    array [2] = 3;

    for (i = 3; i < size; i++)
        array [i] = i * 2;

    return sum (array, size);
}
```

FIG._18A

```
// Global macro definitions
`define UNIT_MAIN        16'h5000
`define UNIT_SIZE        16
`define ADDRESS_SIZE  16
`define DATA_SIZE        32

// Local macro definitions
`define STATE_SIZE       5
`define WORD_SIZE        32
`define WORD_MEMORY_SIZE  15
`define MEMORY_SIZE  480

// External functions defined in the translated C file
`define _sum                    32'h50000001
`define offset__sum             1
`define var__sum                memory [1]
`define state__sum              5'd1
`define _sum__array             32'h50000002
`define offset__sum__array  2
`define var__sum__array      memory [2]
`define _sum__size              32'h50000003
`define offset__sum__size  3
`define var__sum__size       memory [3]
`define _main                   32'h50000004
`define offset__main            4
`define var__main               memory [4]
`define state__main             5'd2

// Static, auto and register variables located in module's internal memory
`define main_3_main_array   32'h50000005
`define offset_main_3_main_array   5

module main (reset, clock, master, in_unit, in_address, in_data, in_read, in_call, out_unit,
                        out_address, out_data, out_read, out_call );

   input        reset, clock, in_call, in_read;
   output       master, out_read, out_call;
   reg          master, out_read, out_call;

   input  [`UNIT_SIZE    - 1 : 0] in_unit;
   input  [`ADDRESS_SIZE - 1 : 0] in_address;
   input  [`DATA_SIZE    - 1 : 0] in_data;

   output [`UNIT_SIZE    - 1 : 0] out_unit;
   output [`ADDRESS_SIZE - 1 : 0] out_address;
   output [`DATA_SIZE    - 1 : 0] out_data;

   reg    [`UNIT_SIZE    - 1 : 0] out_unit;
   reg    [`ADDRESS_SIZE - 1 : 0] out_address;
   reg    [`DATA_SIZE    - 1 : 0] out_data;

 // Module data
   reg    [`STATE_SIZE  - 1 : 0]  state;
   reg    [`STATE_SIZE  - 1 : 0]  return_state;
   reg    [`WORD_SIZE   - 1 : 0]  memory [0 : `WORD_MEMORY_SIZE - 1];
```

FIG._18B1

```
// Predefined states parameter [4:0]
     init        = 5'd0,
     idle        = 5'd20,
     read        = 5'd21,
     write       = 5'd22,
     intercall   = 5'd23,
     intercall2  = 5'd24,
     intercall3  = 5'd25,
     answer      = 5'd26,
     answer2     = 5'd27;
// Function result registers
   reg [31:0] result__sum, result__main;
// Static, auto and register variables located on registers
   reg [31:0] main_2_sum_i, main_2_sum_sum, main_3_main_i, main_3_main_size;
// Temporary registers
   reg [31 : 0]    _6_32, _8_32, _9_32, _11_32;        reg [ 0 : 0]     _7_1, _10_1;
always @(posedge reset or posedge clock) //  Clock cycle
begin
      if (reset)
      begin
         master = 0;
         state  = idle;
      end
      else
      begin
         case (state)
        idle: begin
          if (in_unit == `UNIT_MAIN)
             begin
                if (in_call)
                begin
                   out_unit = in_unit;
                   out_address = in_address;
                   out_data = answer;
                   return_state  = in_address;
                   state = write;
                end
                else
                begin
                   out_unit  = in_unit;
                   out_address  = in_address;
                   return_state  = answer;
                   if (in_read)
                      state = read;
                   else
                   begin
                     out_data  = in_data;
                     state = write;
                   end
                end
              end
            end
```

FIG._18B2

```
init: begin
   master = 1;
   state = answer;
end

`state__sum: begin
   main_2_sum_sum = 32'd0;
   main_2_sum_i = 32'd0;
   state = `STATE_SIZE'd11;
end

`STATE_SIZE'd11: begin
   _6_32 = (main_2_sum_i - `var__sum__size);
   _7_1 = _6_32 [31] == 1'b1;
   state = _7_1 ? `STATE_SIZE'd12 : `STATE_SIZE'd13;
end

`STATE_SIZE'd12: begin
   { out_unit, out_address } = (`var__sum__array + main_2_sum_i);
   return_state = `STATE_SIZE'd17;
   state = read;
end

`STATE_SIZE'd17: begin
   _8_32 = out_data;
   main_2_sum_sum = (main_2_sum_sum + _8_32);
   main_2_sum_i = (main_2_sum_i + 32'd1);
   state = `STATE_SIZE'd11;
end

`STATE_SIZE'd13: begin
   result__sum = main_2_sum_sum;
   { out_unit, out_address } = `_sum;
   out_data = result__sum;
   master = 1;
   state = `var__sum;
end

`state__main: begin
   main_3_main_size = 32'd10;
   memory [(`main_3_main_array - (`UNIT_MAIN << `ADDRESS_SIZE))] = 32'd1;
   memory [(`main_3_main_array + 32'd1) - (`UNIT_MAIN << `ADDRESS_SIZE)] = 32'd2;
   memory [(`main_3_main_array + 32'd2) - (`UNIT_MAIN << `ADDRESS_SIZE)] = 32'd3;
   main_3_main_i = 32'd3;
   state = `STATE_SIZE'd14;
end

`STATE_SIZE'd14: begin
   _9_32 = (main_3_main_i - main_3_main_size);
   _10_1 = _9_32 [31] == 1'b1;
   state = _10_1 ? `STATE_SIZE'd15 : `STATE_SIZE'd16;
end
```

FIG._18B3

```
`STATE_SIZE'd15: begin
    _11_32 = (main_3_main_i << 1'd1);
   { out_unit, out_address } = (`main_3_main_array + main_3_main_i);
    out_data = _11_32;
   return_state = `STATE_SIZE'd18;
    state = write;
end

`STATE_SIZE'd18: begin
    main_3_main_i = (main_3_main_i + 32'd1);
    state = `STATE_SIZE'd14;
end

`STATE_SIZE'd16: begin
   `var__sum__array = `main_3_main_array;
    `var__sum__size = main_3_main_size;
    `var__sum = `STATE_SIZE'd19;
    state = `state__sum;
end

`STATE_SIZE'd19: begin
    result__main = result__sum;
    { out_unit, out_address } = `_main;
    out_data = result__main;
    master = 1;
    state = `var__main;
end

read: begin
    if (out_unit != `UNIT_MAIN)
     begin
        out_read = 1;
        out_call = 0;
        master   = 1;
        state = intercall;
     end
     else
     begin
       out_data  = memory [out_address];
       if (return_state == answer)
            master = 1;
        state = return_state;
     end
end
```

FIG._18B4

```
        write: begin
          if (out_unit != `UNIT_MAIN)
           begin
             out_read = 0;
             out_call = 0;
             master   = 1;
             state  = intercall;
           end
           else
           begin
             memory [out_address] = out_data;
             if (return_state == answer)
                 master = 1;
              state = return_state;
           end
        end

        intercall: begin
           state = intercall2;
        end

        intercall2: begin
           master = 0;
           state  = intercall3;
        end

        intercall3: begin
          if ({ in_unit, in_address } == { out_unit, out_address })
           begin
             out_data  = in_data;
              state = return_state;
           end
        end

        answer: begin
           state = answer2;
        end

        answer2: begin
           master = 0;
           state  = idle;
        end

        default:
           state = idle;
        endcase
      end
    end
endmodule
```

FIG._18B5

```
int data_in;
int out1;
int out2;
int data_out;

void pipeline_stage_1 ()
{
   while (data_in != 0)
      out1 = data_in + 1;

   if (data_in == 1)
      while (data_in == 1)
         out1 = 2;
}

void pipeline_stage_2 ()
{
   while (out1 == 0)
      out2 = 0;

   while (out1 != 0)
      out2 = out1 + 1;
}

void pipeline_stage_3 ()
{
   if (out2 == 1)
      while (out2 == 1)
         data_out = 2;

   while (out2 != 0)
      data_out = out2 + 1;
}

#ifndef __SYNETRY__

#include <stdio.h>

void main ()
{
   for (;;)
   {
      scanf ("%d", & data_in);
      pipeline_stage_1 ();
      pipeline_stage_2 ();
      pipeline_stage_3 ();
      printf ("%d", data_out);
   }
}

#endif
```

FIG._19A

```
module Pipeline(clock,reset,_data_in,_data_out,run_pipeline_stage_1,ready_pipeline_stage_1,
        run_pipeline_stage_2,ready_pipeline_stage_2,run_pipeline_stage_3,ready_pipeline_stage_3);
   input          clock, reset, run_pipeline_stage_1, run_pipeline_stage_2, run_pipeline_stage_3;
   input  [15:0]  _data_in;
   output         ready_pipeline_stage_1, ready_pipeline_stage_2, ready_pipeline_stage_3;
   output [15:0]  _data_out;
   reg            ready_pipeline_stage_1, ready_pipeline_stage_2, ready_pipeline_stage_3;
   reg    [ 2:0]  state1, state2, state3;
   reg    [15:0]  _data_out, _out1, _out2;

   always @(posedge clock)
   begin
      if (reset)
         state1 = 0;
      else
      begin
         case (state1)

          0: begin
             if (run_pipeline_stage_1)
             begin
                ready_pipeline_stage_1 = 0;
                 state1 = 1;
             end
          end

          1: state1 = (_data_in != 16'd0) ? 2 : 3;

          2: begin
             _out1 = (_data_in + 16'd1);
             state1 = 1;
          end

          3: state1 = (_data_in == 16'd1) ? 4 : 6;

          4: state1 = (_data_in == 16'd1) ? 5 : 6;

          5: begin
             _out1 = 16'd2;
             state1 = 4;
          end

          6: begin
            ready_pipeline_stage_1 = 1;
             state1 = 0;
          end

          default:  ;

         endcase
      end
   end
```

FIG._19B1

```
always @(posedge clock)
 begin
    if (reset)
       state2 = 0;
    else
    begin
       case (state2)

        0: begin
           if (run_pipeline_stage_2)
            begin
              ready_pipeline_stage_2 = 0;
               state2 = 1;
            end
         end

        1: state2 = (_out1 == 16'd0) ? 2 : 3;

         2: begin
            _out2 = 16'd0;
            state2 = 1;
         end

         3: state2 = (_out1 != 16'd0) ? 4 : 5;

         4: begin
            _out2 = (_out1 + 16'd1);
            state2 = 3;
         end

         5: begin
           ready_pipeline_stage_2 = 1;
            state2 = 0;
         end

         default:  ;

        endcase
      end
    end
```

FIG._19B2

```
always @(posedge clock)
 begin
    if (reset)
       state3 = 0;
    else
    begin
       case (state3)

        0: begin
           if (run_pipeline_stage_3)
           begin
              ready_pipeline_stage_3 = 0;
              state3 = (_out2 == 16'd1) ? 1 : 3;
           end
        end

        1: state3 = (_out2 == 16'd1) ? 2 : 3;

        2: begin
           _data_out = 16'd2;
           state3 = 1;
        end

        3: state3 = (_out2 != 16'd0) ? 4 : 5;

        4: begin
           _data_out = (_out2 + 16'd1);
           state3 = 3;
        end

        5: begin
           ready_pipeline_stage_3 = 1;
           state3 = 0;
        end

        default: ;

       endcase
    end
  end

endmodule
```

FIG._19B3

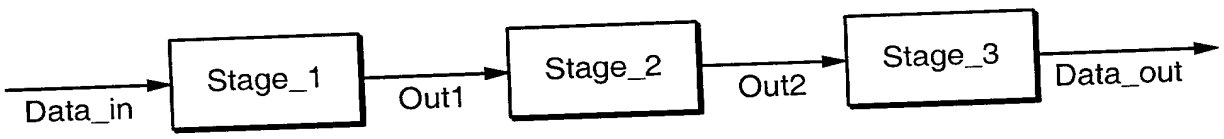

FIG._19C

```
#include <stdio.h>

typedef enum { true = 1, false = 0 } bool;

void main ()
{
   bool bLightIsOn = true;

   printf ("Light.c - C program working on APS-X84 FPGA test card\n");

   for (;;)
   {
      int counter;

      printf
      (
         "Enter the number of clock ticks"
         " to wait before the light change: "
      );

      scanf ("%d", & counter);

      if (counter < 0)
         counter = 0;

      while (counter -- != 0)
         printf ("Waiting...\n");

      bLightIsOn = ! bLightIsOn;

      printf ("Light is: %d\n", bLightIsOn);
   }
}
```

FIG._20A

```
module Light (clock, reset, scanf_0_1_line_33_counter, printf_4_1_line_43_bLightIsOn,
                run_main, ready_main );
  input          clock, reset, run_main;
  input  [ 7:0]  scanf_0_1_line_33_counter;

  output         ready_main;
  output [ 1:0]  printf_4_1_line_43_bLightIsOn;
  reg            ready_main, _1_1;
  reg    [ 1:0]  printf_4_1_line_43_bLightIsOn, _3_main_bLightIsOn, 3_2;
  reg    [ 2:0]  state;
  reg    [ 7:0]  _4_main_counter, _2_8;

  always @(posedge clock)
  begin
    if (reset)
      state = 0;
    else
    begin
      case (state)

      0: begin
        if (run_main)
        begin
          ready_main = 0;
          _3_main_bLightIsOn = 2'd1;
          // User Verilog code
          $write ("Light.c - C program working on APS-X84 FPGA test card\n");
          // End of user Verilog code
          state = 1;
        end
      end

      1: begin
        // User Verilog code
        $write ("Enter the number of clock ticks to wait before the light change: ");
        // End of user Verilog code
        _4_main_counter = scanf_0_1_line_33_counter;
        _1_1 = _4_main_counter [7] == 1'b1;
        if (_1_1)
          _4_main_counter = 8'd0;
        state = 2;
      end

      2: begin
        _2_8 = _4_main_counter;
        _4_main_counter = (_2_8 - 8'd1);
        state = (_2_8 != 8'd0) ? 3 : 4;
      end
```

FIG._20B1

```
            3: begin
               // User Verilog code
                $write ("Waiting...\n");
               // End of user Verilog code
                state = 2;
            end

            4: begin
               _3_2 = (! _3_main_bLightIsOn);
               _3_main_bLightIsOn = _3_2;
               printf_4_1_line_43_bLightIsOn = _3_main_bLightIsOn;
               // User Verilog code
               $write ("Light is: %d\n", printf_4_1_line_43_bLightIsOn);
               // End of user Verilog code
                state = 1;
            end

            default: ;

           endcase
        end
     end

endmodule

//     APSX84.v - Verilog wrapper module which instantiates module Light.v.
//              This design is intended to run on APS-X84 board.
//

module APSX84 ( clock, clock_enable, reset, data, LCD);

    input          clock       /* synthesis xc_loc=P24  */ ;
    input          clock_enable /* synthesis xc_loc=P3   */ ;
    input          reset       /* synthesis xc_loc=P4   */ ;
    input  [5:0]   data        /* synthesis xc_loc="P10,P9,P8,P7,P6,P5" */ ;
    output         LCD          /* synthesis xc_loc=P35  */ ;

    wire           internal_clock = clock & clock_enable;
    wire   [7:0]   data8 = { data, 2'b0 };
    wire   [1:0]   LCD2;
    wire           LCD = LCD2 [0];
    wire           run_main  = 1;

    Light Light ( internal_clock, reset, data8, LCD2, run_main,);

endmodule
```

FIG._20B2

```
int sum (int n)
{
   int i, sum = 0;
   for (i = 0; i != n; i++)
      sum += i;
   return sum;
}
```

FIG._21A

```
module Sum (clock, reset, result__sum, param__sum__n, run__sum, ready__sum );
   input          clock, reset, run__sum;
   input  [15:0]  param__sum__n;
   output         ready__sum;
   output [15:0]  result__sum;
   reg            ready__sum;
   reg    [ 1:0]  state;
   reg    [15:0]  result__sum, s_2_sum_i, s_2_sum_sum;
   always @(posedge clock)
   begin
      if (reset)
         state = 0;
      else
      begin
         case (state)
          0: begin
             if (run__sum)
             begin
                ready__sum = 0;
                s_2_sum_sum = 16'd0;
                s_2_sum_i = 16'd0;
                state = 1;
             end
          end
          1: state = (s_2_sum_i != param__sum__n) ? 2 : 3;
          2: begin
              s_2_sum_sum = (s_2_sum_sum + s_2_sum_i);
              s_2_sum_i = (s_2_sum_i + 16'd1);
              state = 1;
          end
          3: begin
              result__sum = s_2_sum_sum;
             ready__sum = 1;
              state = 0;
          end
          default: ;
          endcase
      end
   end
endmodule
```

FIG._21B

```
library IEEE;
use IEEE.STD_Logic_1164.all;
use IEEE.Numeric_Std.all;

entity synetry_sum is
port( clock        : in  std_logic;
      reset        : in  boolean;
      run_sum      : in  boolean;
      param_sum_n : in  integer range 0 to 65535;
      result_sum  : out integer range 0 to 65535;
      ready_sum   : out boolean  );
end entity synetry_sum;

architecture RTL of synetry_sum is
begin
   process_synetry_sum : process
      variable state       : integer range 0 to 3;
      variable s_2_sum_i   : integer range 0 to 65535;
      variable s_2_sum_sum : integer range 0 to 65535;
    begin
      wait until clock'event and clock = '1';
      if (reset) then
         state := 0;
       else
         case (state)is
            when 0 =>
              if (run_sum)then
                 ready_sum <= false;
                  s_2_sum_sum := 0;
                  s_2_sum_i := 0;
                  state := 1;
               end if;
             when 1 =>
               if (s_2_sum_i /= param_sum_n) then
                   state := 2;
                else
                   state := 3;
                end if;
              when 2 =>
                 s_2_sum_sum := s_2_sum_sum + s_2_sum_i;
                 s_2_sum_i := s_2_sum_i + 1;
                 state := 1;
              when 3 =>
                 result_sum <= s_2_sum_sum;
                 ready_sum <= true;
                  state := 0;
           end case;
         end if;
         wait;
      end process process_synetry_sum;
    end architecture RTL;
```

FIG._21C

```
int read (int * p)
{
   return * p;
}
```

FIG._22A

```
module Read(clock,reset,a,d,o,we,result_read,param_read_p,run_read,ready_read);
   input          clock, reset, run_read;
   input  [ 3:0]  o, param_read_p;
   output         we, ready_read;
   output [ 3:0]  a, d, result_read;
   reg            we, ready_read, state;
   reg    [ 3:0]  a, d, result_read;

  always @(posedge clock)
  begin
     if (reset)
     begin
        we = 0;
        state = 0;
     end
     else
     begin
        case (state)

        0:begin
           if (run_read)
            begin
              ready_read = 0;
              a = param_read_p;
               state = 1;
           end
        end

        1:begin
           _1_4 = o;
          result_read = _1_4;
          ready_read = 1;
           state = 0;
        end

        default:  ;
        endcase
     end
  end
endmodule
```

FIG._22B

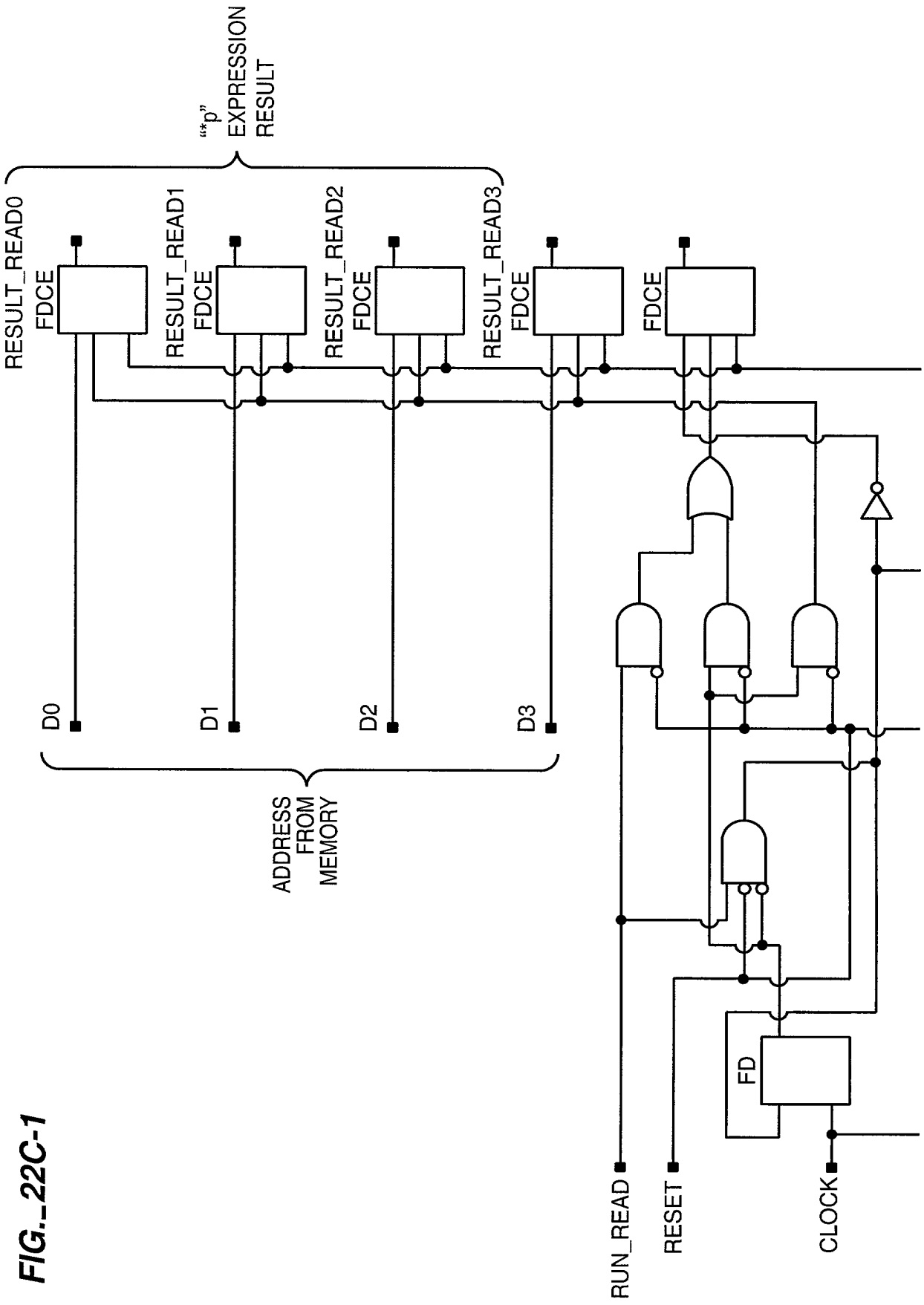
"*p" EXPRESSION RESULT
RESULT_READ0
RESULT_READ1
RESULT_READ2
RESULT_READ3
FDCE
FDCE
FDCE
FDCE
FDCE
D0
D1
D2
D3
ADDRESS FROM MEMORY
FD
RUN_READ
RESET
CLOCK

FIG._22C-1

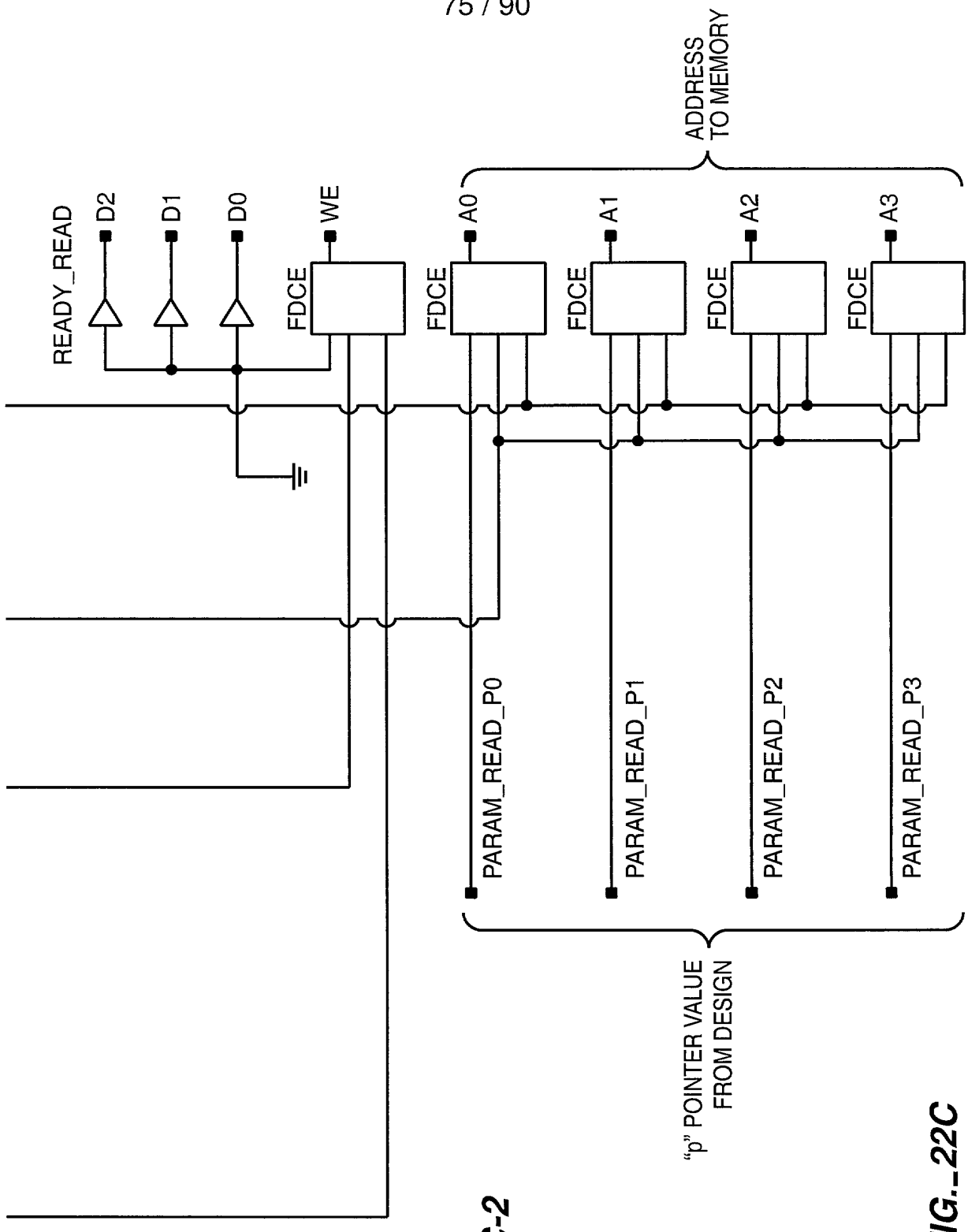

FIG._22C-2

FIG._22C-1

FIG._22C-2

FIG._22C

```
void write (int * p, int d)
{
   * p = d;
}
```

FIG._23A

```
module Write(clock,reset,a,d,o,we,param_write_p,param_write_d,run_write,ready_write);
   input           clock, reset, run_write;
   input   [ 3:0]  o, param_write_p, param_write_d;
   output          we, ready_write;
   output  [ 3:0]  a, d;
   reg             we, ready_write, state;
   reg     [ 3:0]  a, d;

   always @(posedge clock)
   begin
     if (reset)
     begin
       we = 0;
       state = 0;
     end
     else
     begin
       case (state)

       0:begin
         if (run_write)
         begin
           ready_write = 0;
           a = param_write_p;
           d = param_write_d;
           we = 1;
           state = 1;
         end
       end

       1:begin
         we = 0;
         ready_write = 1;
         state = 0;
       end

       default:   ;

       endcase
     end
   end

 endmodule
```

FIG._23B

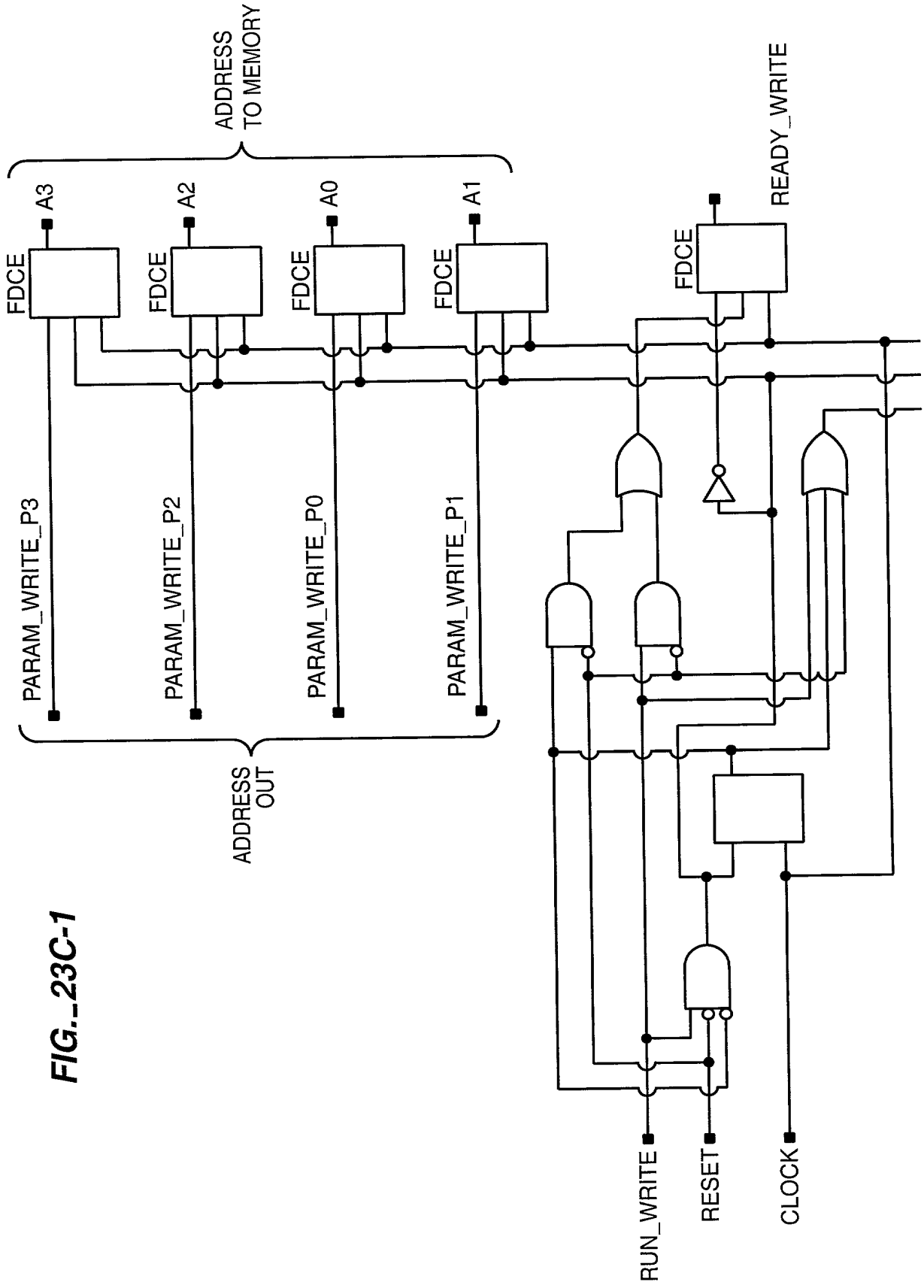
ADDRESS TO MEMORY
A3
A2
A0
A1
FDCE
FDCE
FDCE
FDCE
FDCE
READY_WRITE
PARAM_WRITE_P3
PARAM_WRITE_P2
PARAM_WRITE_P0
PARAM_WRITE_P1
ADDRESS OUT
RUN_WRITE
RESET
CLOCK

FIG._23C-1

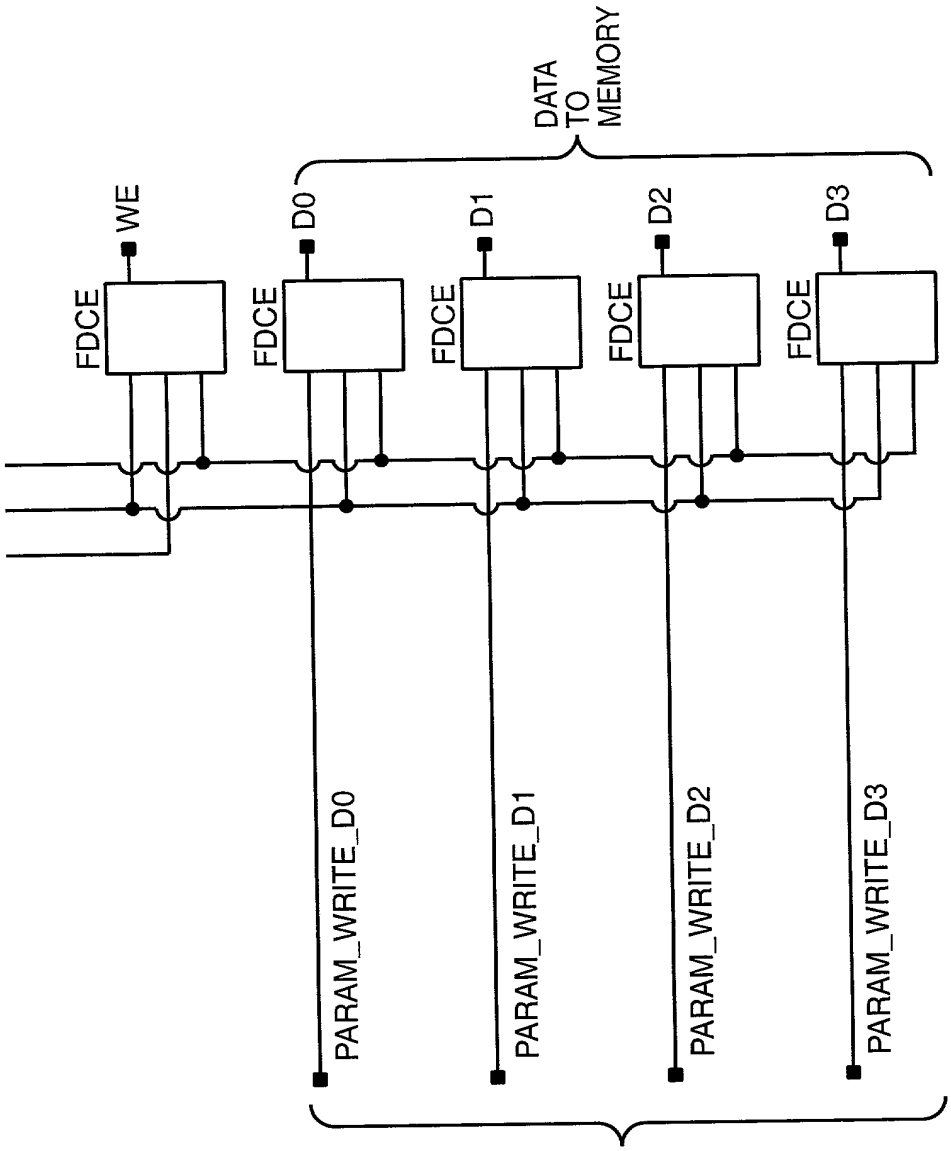

FIG._23C-2

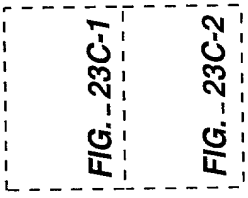

FIG._23C

```
typedef struct
{
        int a;
        int b;
} STRUCTURE;

int read_member_b (STRUCTURE *p)
{
   return p -> b;
}
```

FIG._24A

```
module Struct1(clock,reset,a,d,o,we,result_read_member_b,param_read_member_b_p,
                    run_read_member_b,ready_read_member_b);
   input          clock, reset, run_read_member_b;
   input  [ 3:0]  o, param_read_member_b_p;
   output         we, ready_read_member_b;
   output [ 3:0]  a, d, result_read_member_b;
   reg            we, ready_read_member_b, state;
   reg    [ 3:0]  a, d, result_read_member_b, _1_4;

   always @(posedge clock)
   begin
      if (reset)
      begin
         we = 0;
         state = 0;
      end
      else
      begin
         case (state)

          0:begin
            if (run_read_member_b)
             begin
               ready_read_member_b = 0;
               a = (param_read_member_b_p + 4'd1);
                state = 1;
             end
          end

          1:begin
             _1_4 = o;
            result_read_member_b = _1_4;
            ready_read_member_b = 1;
             state = 0;
          end

          default:  ;

          endcase
       end
    end
endmodule
```

FIG._24B

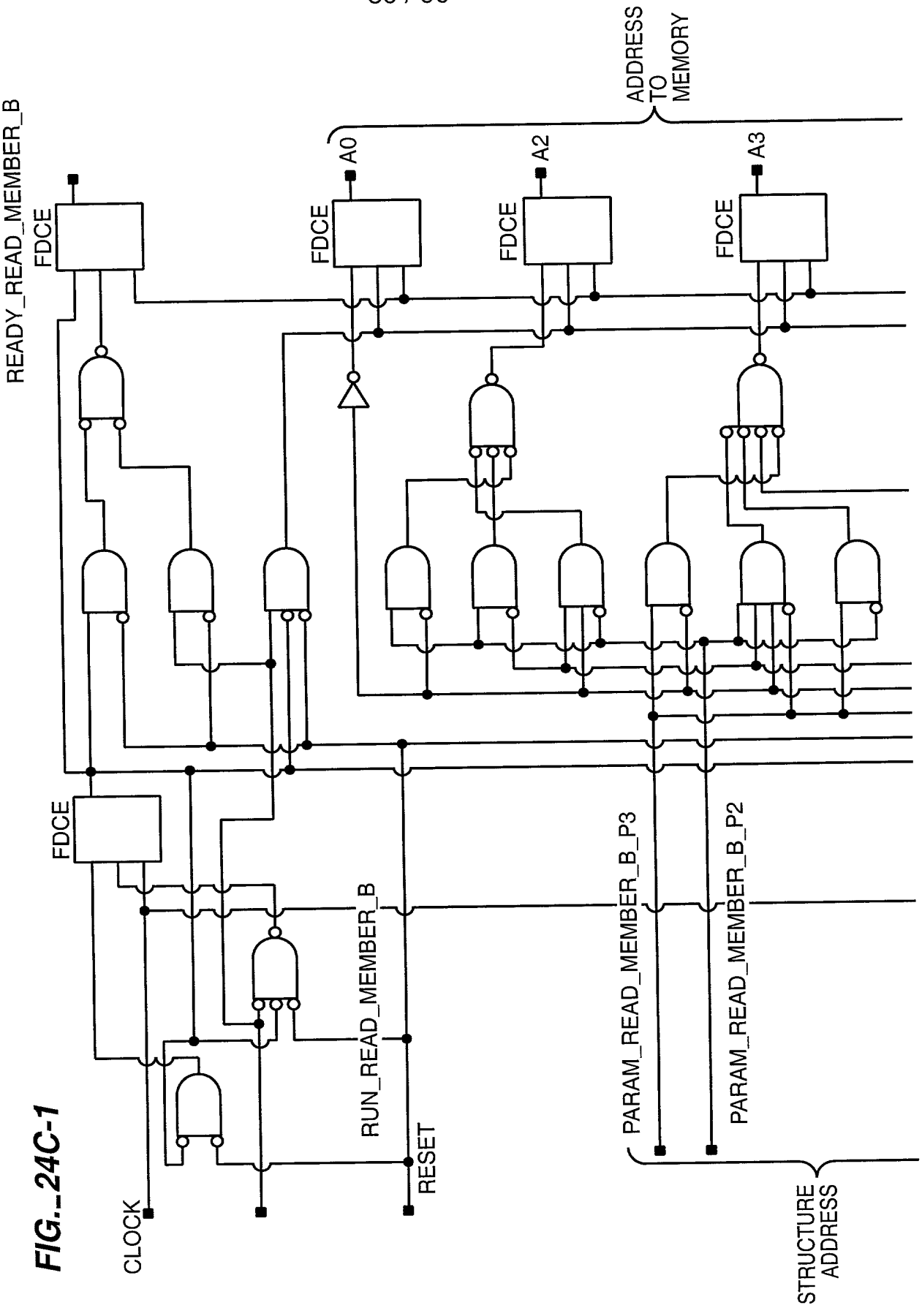

FIG._24C-1

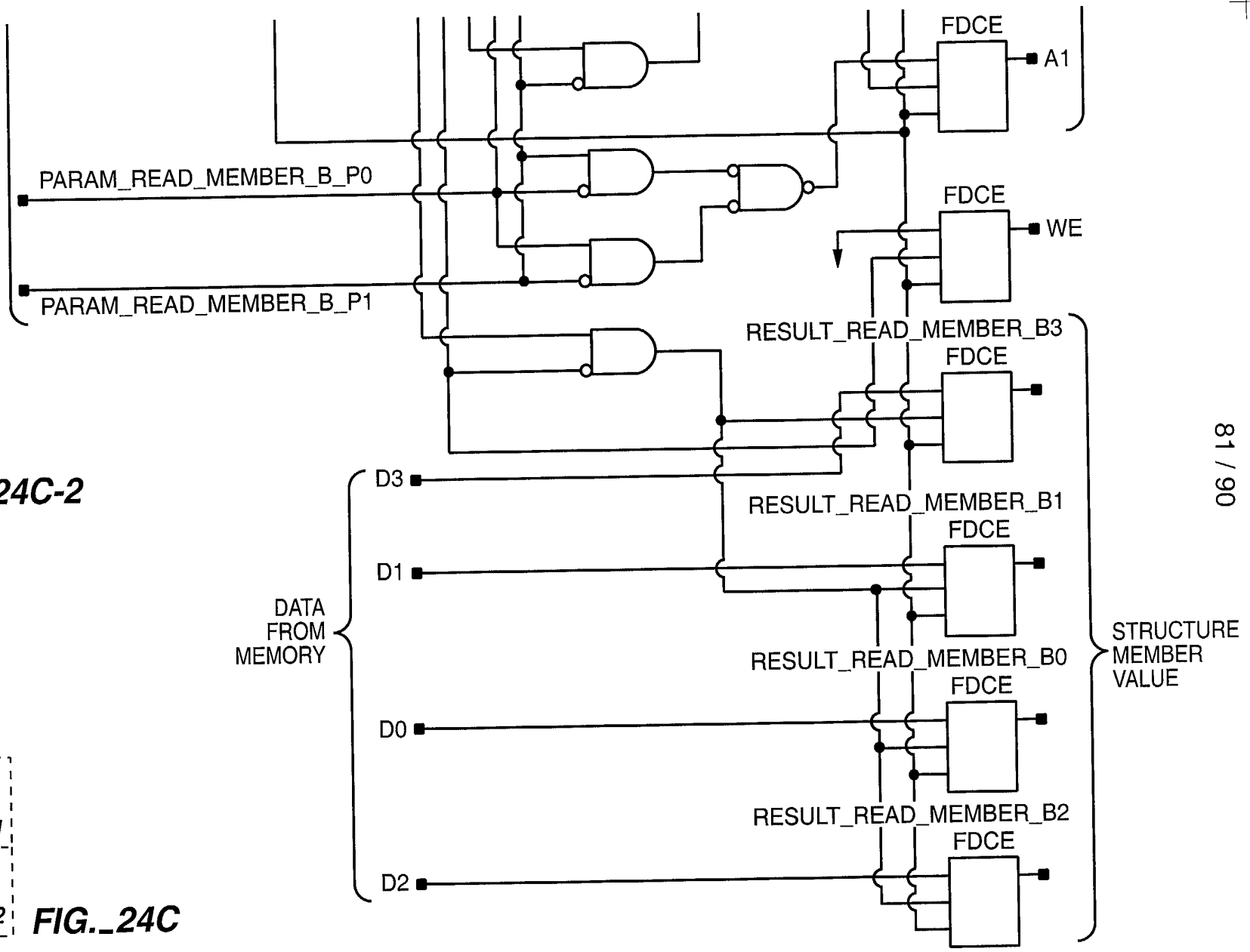

FIG._24C-2

FIG._24C-1

FIG._24C-2

FIG._24C

```
typedef struct
{
        int a;
        int b;
} STRUCTURE;

void write_member_b (STRUCTURE *p, int n)

{
   p -> b = n;
}
```

FIG._25A

```
module Struct2(clock,reset,a,d,o,we,param_write_member_b_p,param_write_member_b_n,
                        run_write_member_b,ready_write_member_b);
   input          clock, reset, run_write_member_b;
   input  [ 3:0]  o, param_write_member_b_p, param_write_member_b_n;
   output         we, ready_write_member_b;
   output [ 3:0]  a, d;
   reg            we, ready_write_member_b, ready_write_member_b, state;
   reg    [ 3:0]  a, d;

   always @(posedge clock)
   begin
      if (reset)
      begin
         we = 0;
         state = 0;
      end
      else
      begin
         case (state)

          0:begin
             if (run_write_member_b)
             begin
                ready_write_member_b = 0;
                a = (param_write_member_b_p + 4'd1);
                d = param_write_member_b_n;
                 we = 1;
                 state = 1;
             end
          end

          1:begin
             we = 0;
            ready_write_member_b = 1;
             state = 0;
          end

          default:  ;

         endcase
      end
   end
endmodule
```

FIG._25B

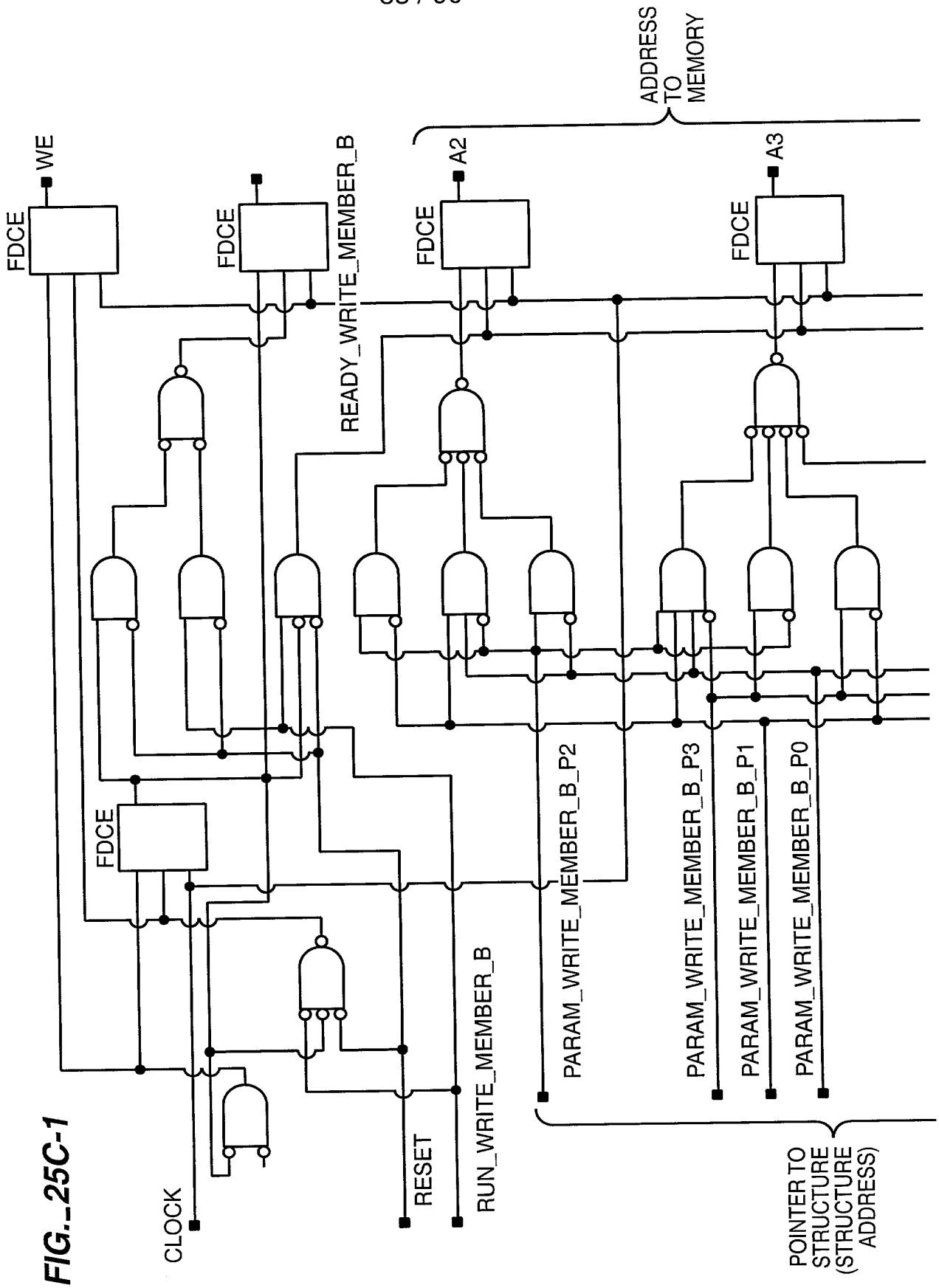
WE
FDCE
FDCE
READY_WRITE_MEMBER_B
FDCE
A2
FDCE
A3
ADDRESS TO MEMORY
FDCE
CLOCK
RESET
RUN_WRITE_MEMBER_B
PARAM_WRITE_MEMBER_B_P2
PARAM_WRITE_MEMBER_B_P3
PARAM_WRITE_MEMBER_B_P1
PARAM_WRITE_MEMBER_B_P0
POINTER TO STRUCTURE (STRUCTURE ADDRESS)

FIG._25C-1

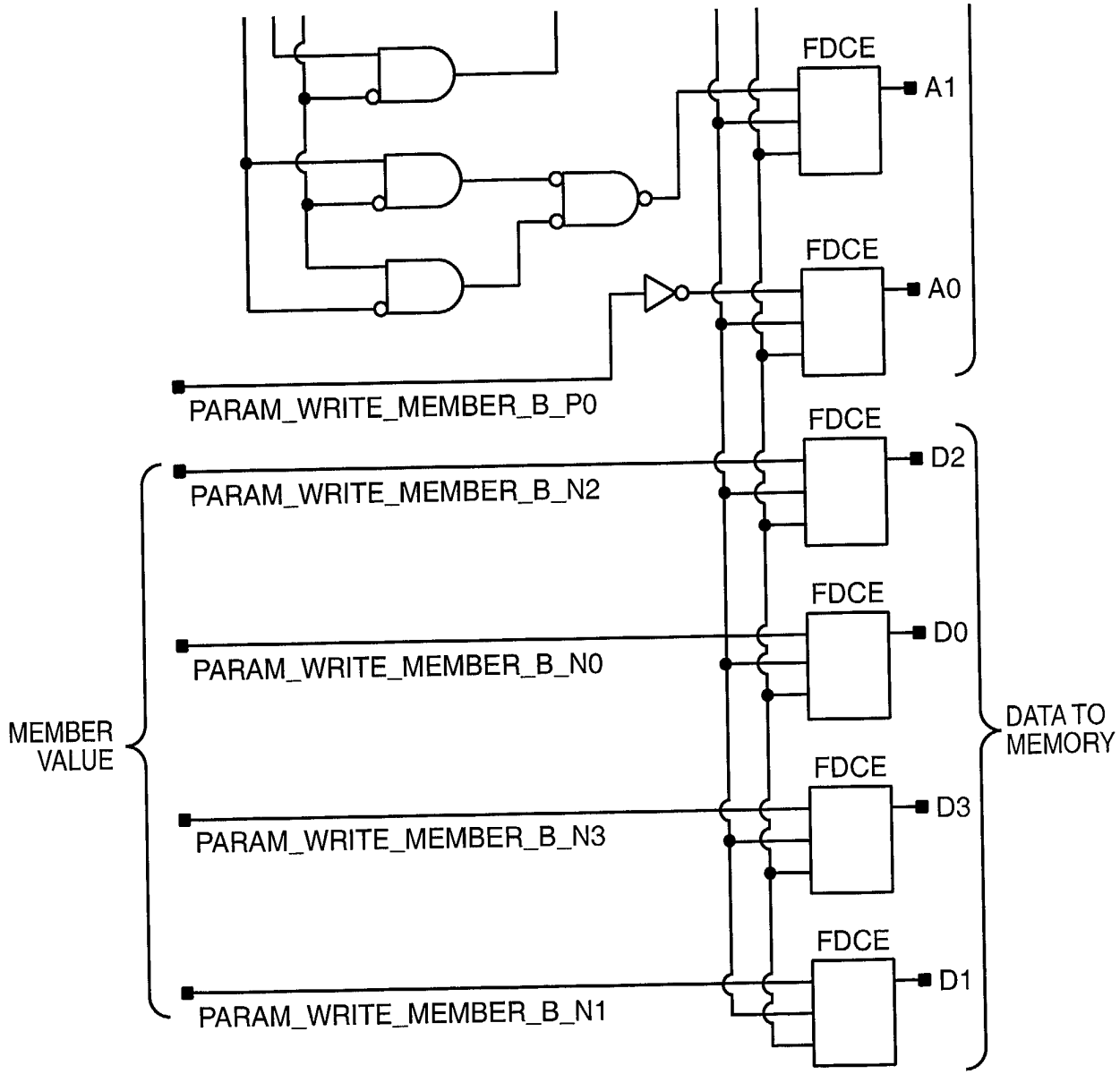

FIG._25C-2

FIG._25C-1

FIG._25C-2

FIG._25C

```
typedef struct
{
        int a;
        int b;
} STRUCTURE;
STRUCTURE s;
int read_member_b (void)
{
  return s.b;
}
```

FIG._26A

```
`define _s 2'h1
module Struct4(clock,reset,a,d,o,we,result_read_member_b,run_read_member_b,ready_read_member_b);
   input          clock, reset, run_read_member_b;
   input  [ 3:0]  o;
   output         we, ready_read_member_b;
   output [ 1:0]  a, d, result_read_member_b;
   reg            we, ready_read_member_b, state;
   reg    [ 1:0]  a, d, result_read_member_b;
   reg    [ 3:0]  _1_4 ;
  always @(posedge clock)
   begin
     if (reset)
     begin
        we = 0;
        state = 0;
     end
     else
     begin
        case (state)
        0: begin
          if (run_read_member_b)
           begin
             ready_read_member_b = 0;
              a = (`_s + 4'd1);
              state = 1;
           end
        end
        1: begin
           _1_4 = o;
          result_read_member_b = _1_4;
          ready_read_member_b = 1;
           state = 0;
        end
        default:  ;
        endcase
     end
  end
endmodule
```

FIG._26B

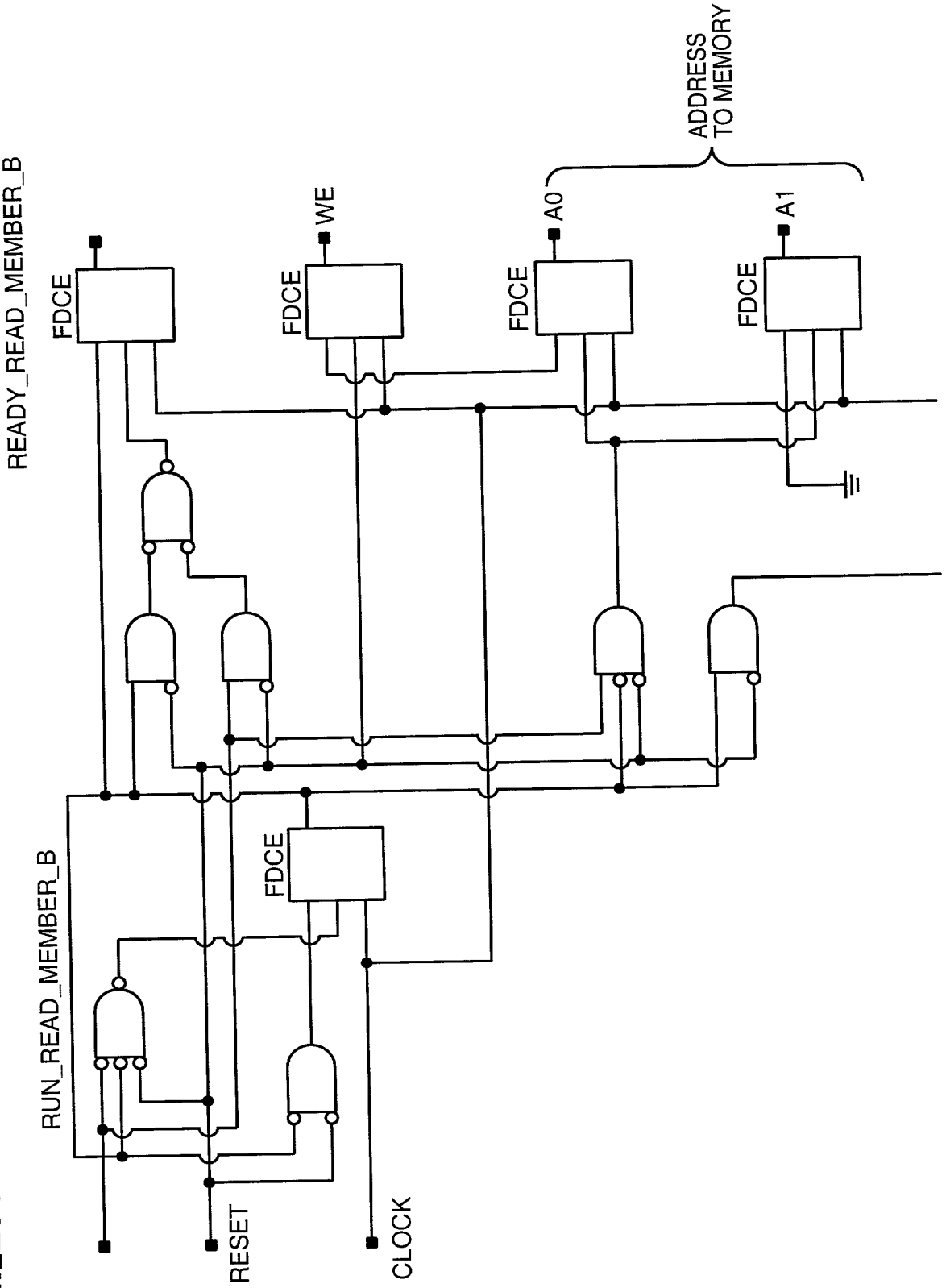

FIG._26C-1

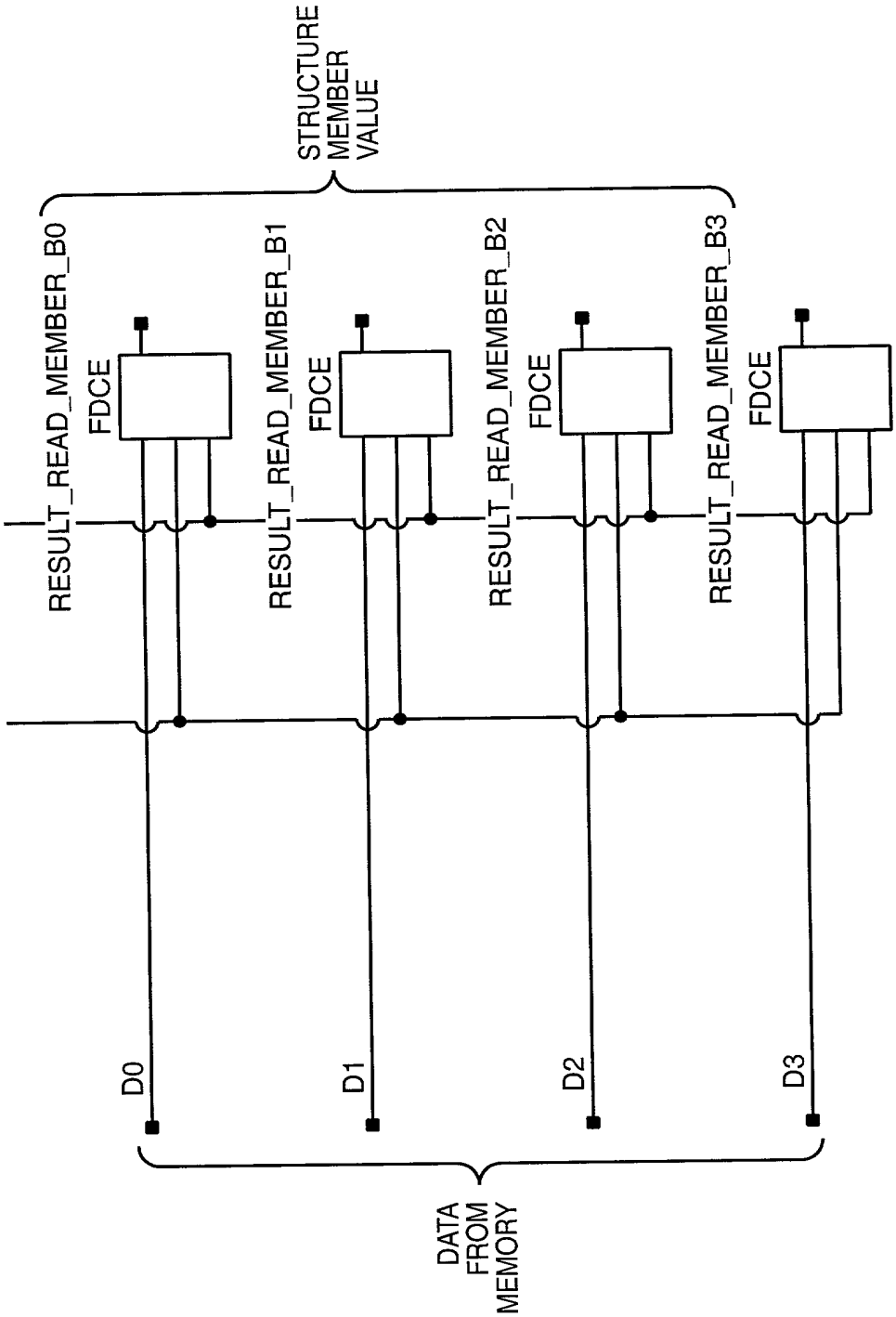

FIG._26C-2

FIG._26C-1

FIG._26C-2

FIG._26C

```
typedef struct
{
        int a;
        int b;
} STRUCTURE;
STRUCTURE s;
void write_member_b (int n)
{
  s.b = n;
}
```

FIG._27A

```
`define _s  2'h1
module Struct5(clock,reset,a,d,o,we,param_write_member_b_n,run_write_member_b,ready_write_member_b);
   input          clock, reset, run_write_member_b;
   input  [ 3:0]  param_write_member_b_n, o;
   output         we, ready_write_member_b;
   output [ 1:0]  a;
   output [ 3:0]  d;
   reg            we, ready_write_member_b, state;
   reg    [ 1:0]  a;
   reg    [ 3:0]  d;
  always @(posedge clock)
   begin
      if (reset)
      begin
         we = 0;
         state = 0;
      end
      else
      begin
        case (state)
       0: begin
            if (run_write_member_b)
            begin
              ready_write_member_b = 0;
               a = (`_s + 4'd1);
              d = param_write_member_b_n;
               we = 1;
               state = 1;
            end
         end
       1: begin
            we = 0;
           ready_write_member_b = 1;
            state = 0;
         end
     default:   ;
     endcase
      end
  end
endmodule
```

FIG._27B

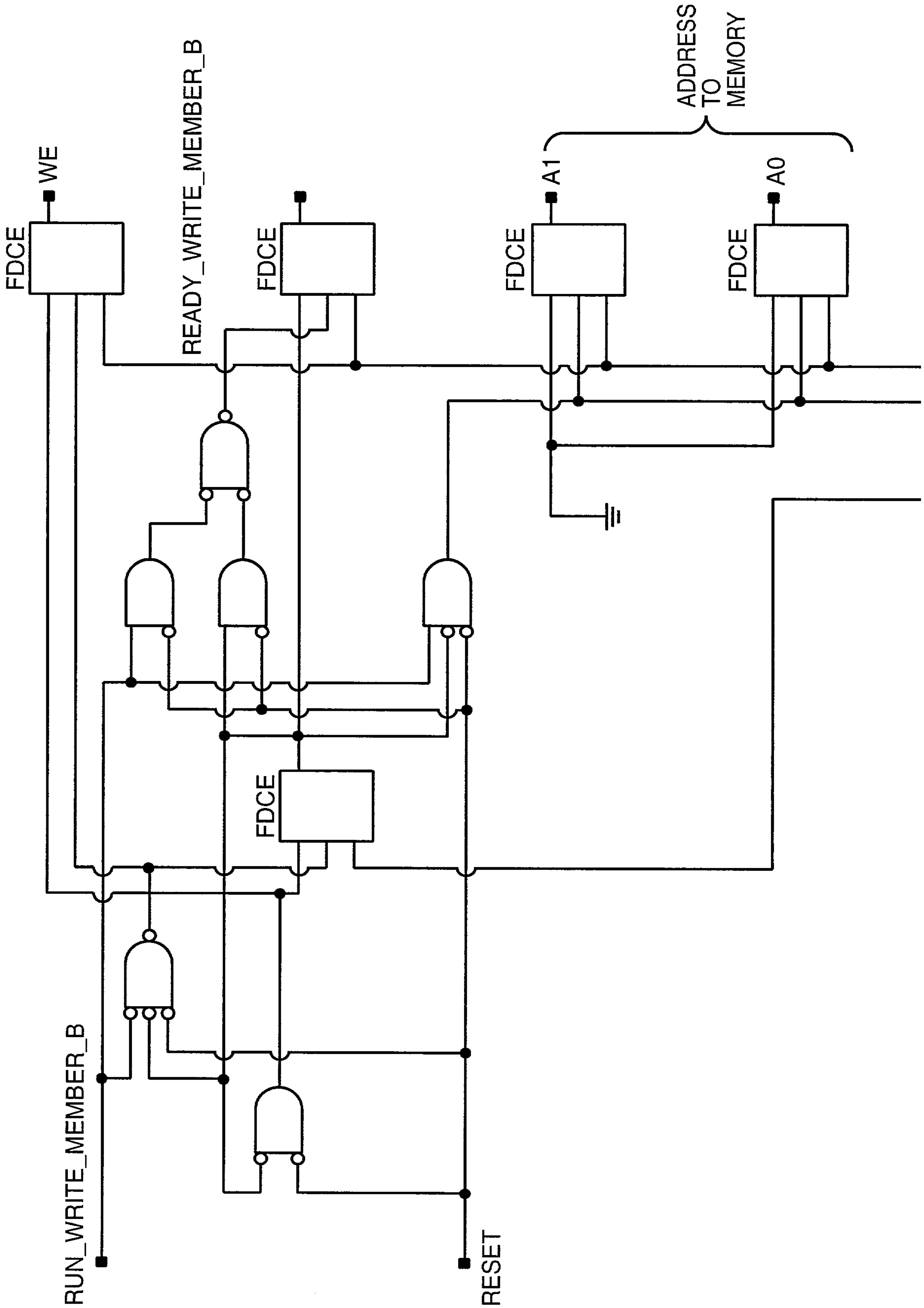
WE
FDCE
FDCE
FDCE
FDCE
FDCE
READY_WRITE_MEMBER_B
A1
A0
ADDRESS
TO
MEMORY
RUN_WRITE_MEMBER_B
RESET

FIG._27C-1

FIG._27C-2

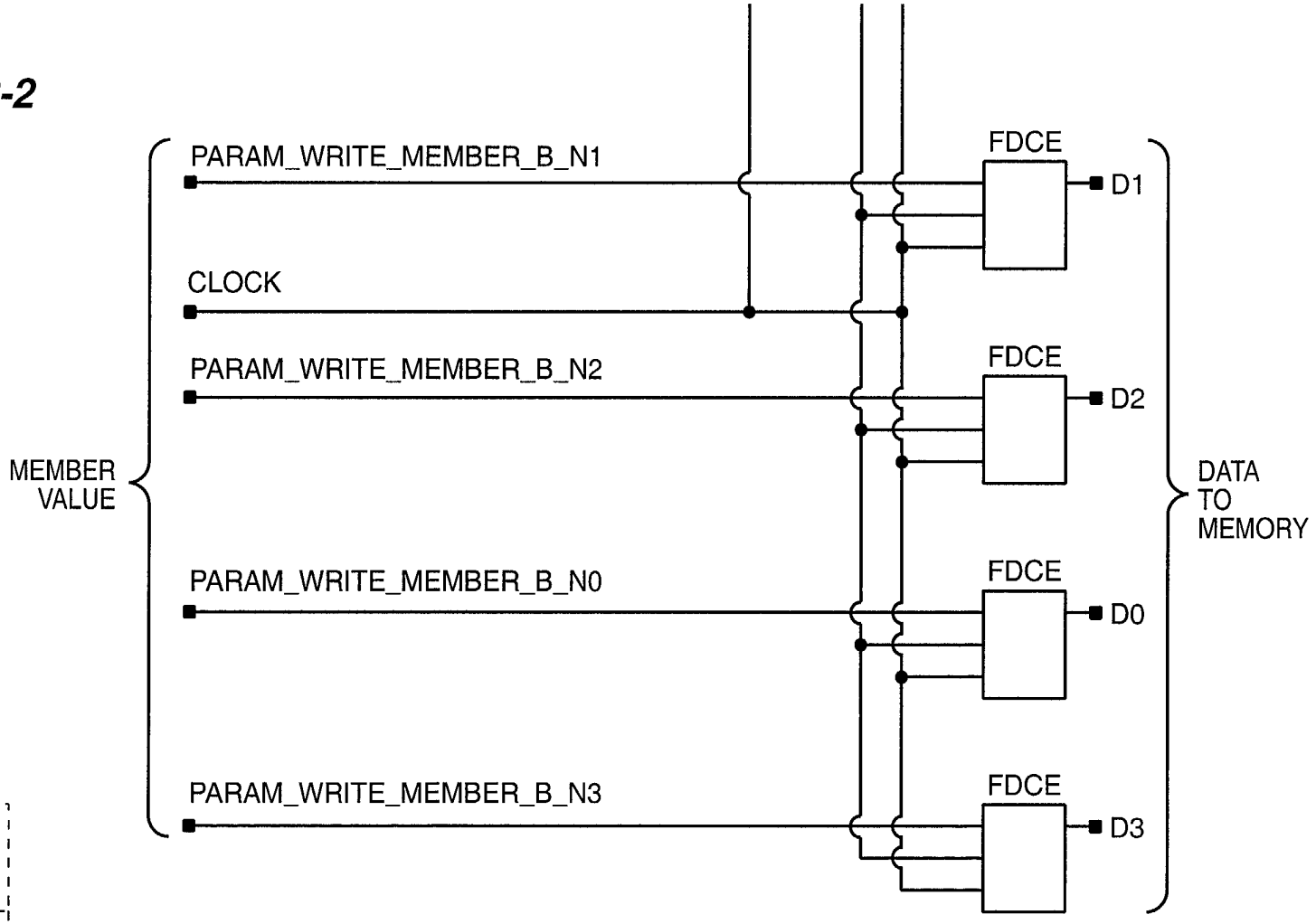

FIG._27C-1

FIG._27C-2

FIG._27C